SHFFT 1 OF 140

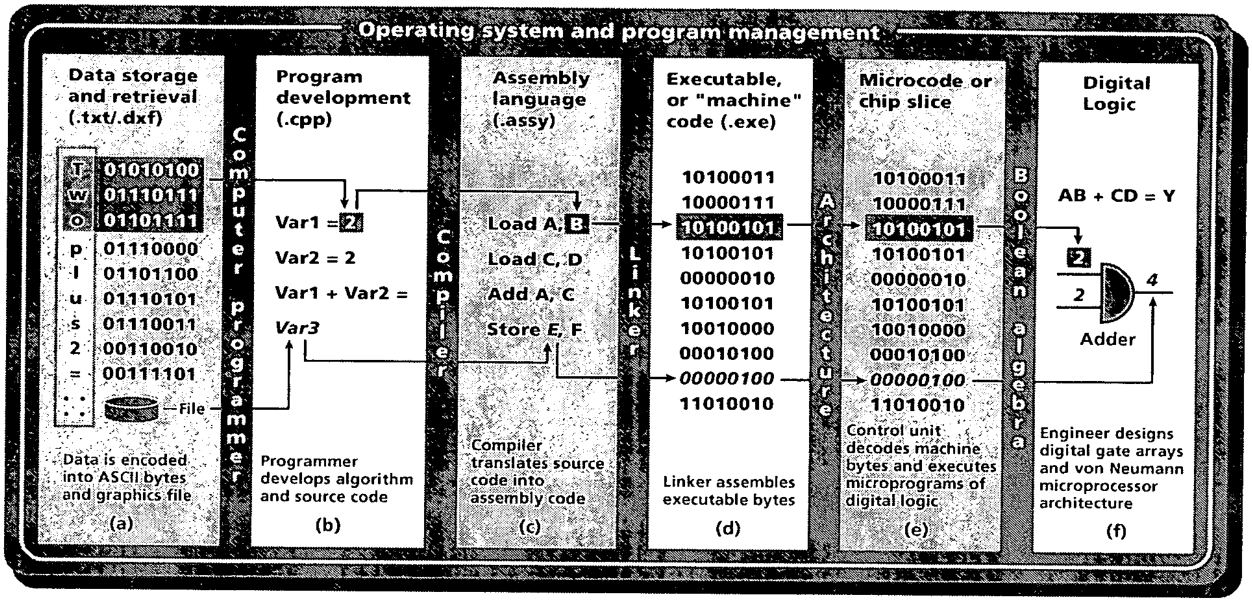
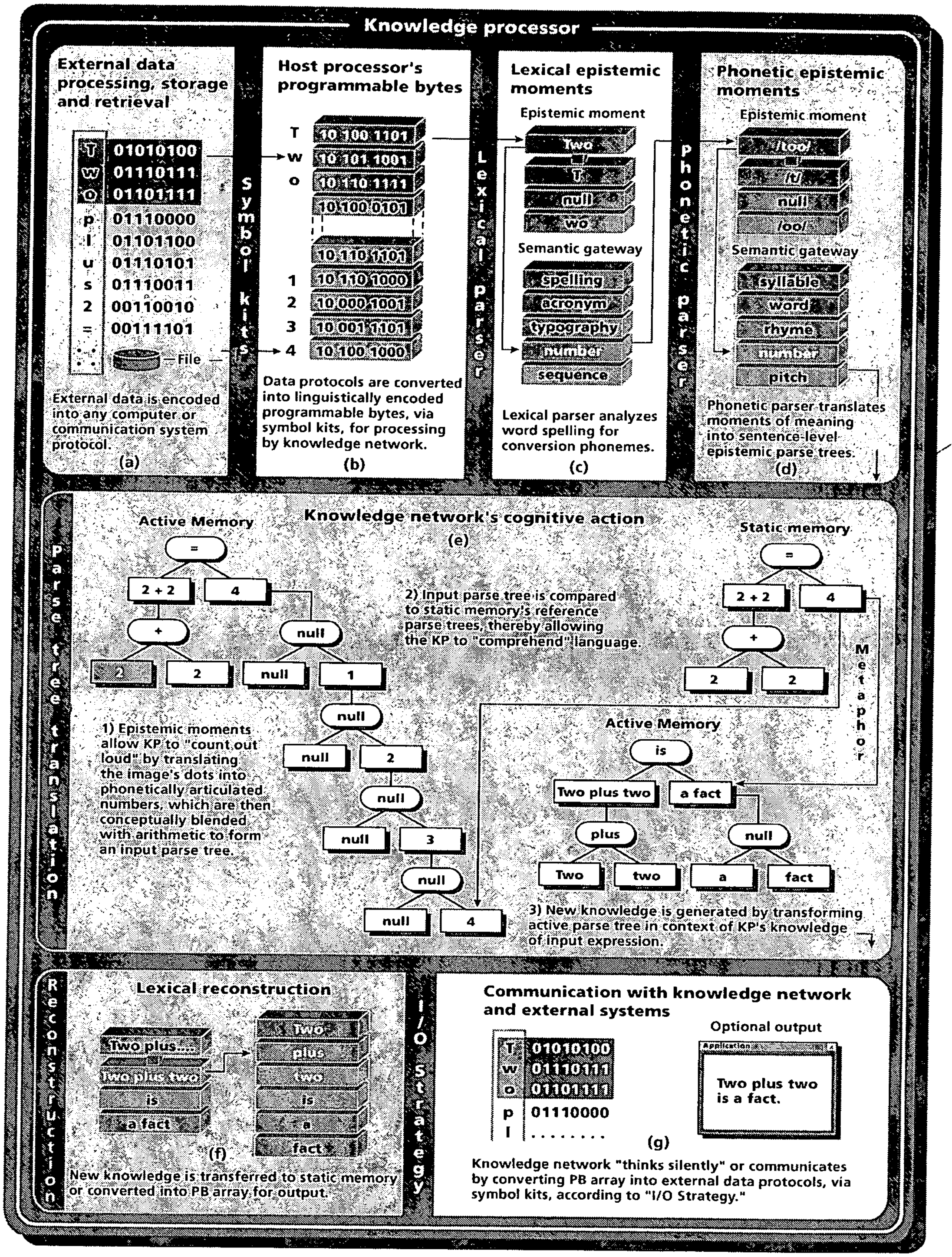

Operating system and program management
Data storage and retrieval (.txt/.dxf)
T W O p l u s 2 =
01010100
01110111
01101111
01110000
01101100
01110101
01110011
00110010
00111101
File
Data is encoded into ASCII bytes and graphics file
(a)
Computer programmer
Program development (.cpp)
Var1 = 2
Var2 = 2
Var1 + Var2 =
Var3
Programmer develops algorithm and source code
(b)
Compiler
Assembly language (.assy)
Load A, B
Load C, D
Add A, C
Store E, F
Compiler translates source code into assembly code
(c)
Linker
Executable, or "machine" code (.exe)
10100011
10000111
10100101
10100101
00000010
10100101
10010000
00010100
00000100
11010010
Linker assembles executable bytes
(d)
Architecture
Microcode or chip slice
10100011
10000111
10100101
10100101
00000010
10100101
10010000
00010100
00000100
11010010
Control unit decodes machine bytes and executes microprograms of digital logic
(e)
Boolean algebra
Digital Logic
AB + CD = Y
2
2
4
Adder
Engineer designs digital gate arrays and von Neumann microprocessor architecture
(f)

Fig.1

SHEET 2 OF 140



External data is encoded into any computer or communication system protocol.

(a)

Data protocols are converted into linguistically encoded programmable bytes, via symbol kits, for processing by knowledge network.

(b)

Lexical parser analyzes word spelling for conversion phonemes.

(c)

Phonetic parser translates moments of meaning into sentence-level epistemic parse trees.

(d)

1) Epistemic moments allow KP to "count out loud" by translating the image's dots into phonetically articulated numbers, which are then conceptually blended with arithmetic to form an input parse tree.

2) Input parse tree is compared to static memory's reference parse trees, thereby allowing the KP to "comprehend" language.

3) New knowledge is generated by transforming active parse tree in context of KP's knowledge of input expression.

New knowledge is transferred to static memory or converted into PB array for output.

(f)

Two plus two is a fact.

(g)

Knowledge network "thinks silently" or communicates by converting PB array into external data protocols, via symbol kits, according to "I/O Strategy."

Fig.2

SHEET 3 OF 140

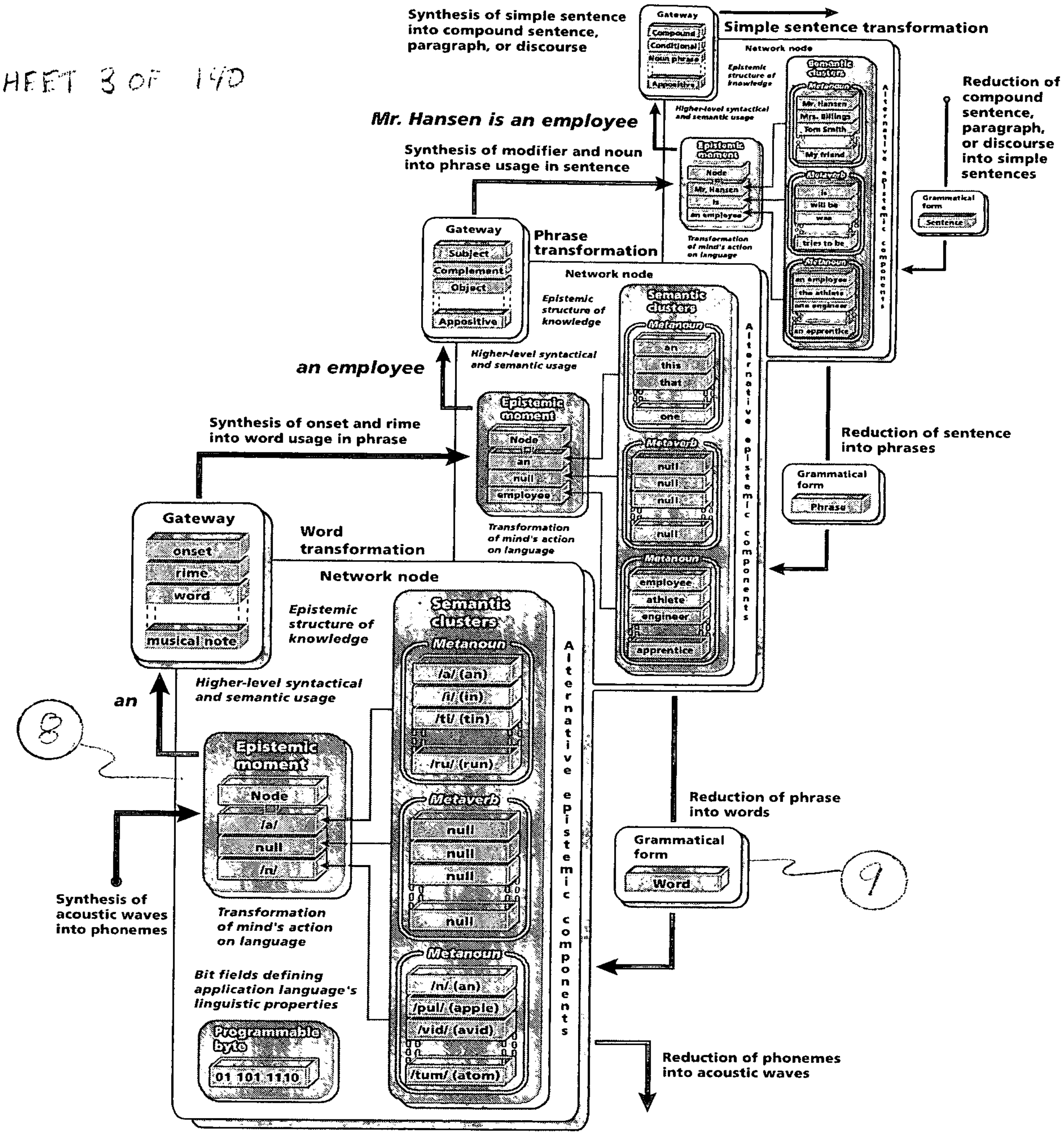
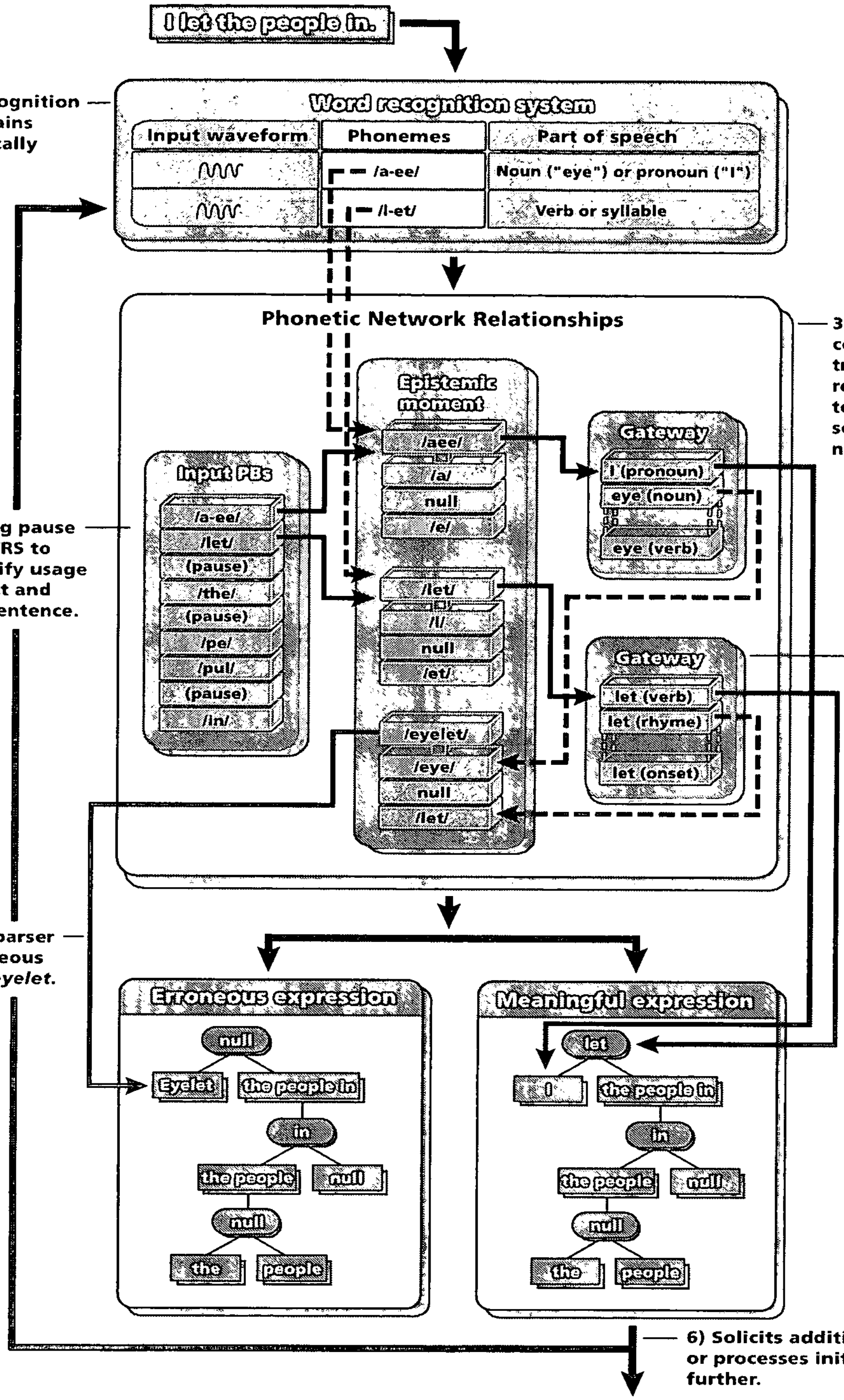

Synthesis of simple sentence into compound sentence, paragraph, or discourse
Gateway
Compound
Conditional
Noun phrase
Appositive
Simple sentence transformation
Network node
Epistemic structure of knowledge
Semantic clusters
Metanoun
Mr. Hansen
Mrs. Billings
Tom Smith
My friend
Metaverb
is
will be
was
tries to be
Metanoun
an employee
the athlete
one engineer
an apprentice
Alternative epistemic components
Reduction of compound sentence, paragraph, or discourse into simple sentences
Grammatical form
Sentence
Higher-level syntactical and semantic usage
Mr. Hansen is an employee
Synthesis of modifier and noun into phrase usage in sentence
Epistemic moment
Node
Mr. Hansen
is
an employee
Transformation of mind's action on language
Gateway
Subject
Complement
Object
Appositive
Phrase transformation
Network node
Epistemic structure of knowledge
Higher-level syntactical and semantic usage
an employee
Semantic clusters
Metanoun
an
this
that
one
Metaverb
null
null
null
null
Metanoun
employee
athlete
engineer
apprentice
Alternative epistemic components
Reduction of sentence into phrases
Grammatical form
Phrase
Synthesis of onset and rime into word usage in phrase
Epistemic moment
Node
an
null
employee
Transformation of mind's action on language
Gateway
onset
rime
word
musical note
Word transformation
Network node
Epistemic structure of knowledge
Higher-level syntactical and semantic usage
an
8
Epistemic moment
Node
/a/
null
/n/
Synthesis of acoustic waves into phonemes
Transformation of mind's action on language
Bit fields defining application language's linguistic properties
Programmable byte
01 101 1110
Semantic clusters
Metanoun
/a/ (an)
/i/ (in)
/ti/ (tin)
/ru/ (run)
Metaverb
null
null
null
null
Metanoun
/n/ (an)
/pul/ (apple)
/vid/ (avid)
/tum/ (atom)
Alternative epistemic components
Reduction of phrase into words
Grammatical form
Word
9
Reduction of phonemes into acoustic waves

Fig.3

SHEET 4 OF 140



1) Word recognition system obtains mathematically defined phonemes.

2) Missing pause causes WRS to misidentify usage of subject and verb in sentence.

3) Knowledge network compares input parse tree to static memory's reference parse trees to formulate potential sentences using network gateways.

4) Language parser detects erroneous use of word *eyelet*.

5) Knowledge network determines waveforms' correct parts of speech by formulating meaningful sentence in universal grammar.

6) Solicits additional input or processes initial expression further.

Fig.4

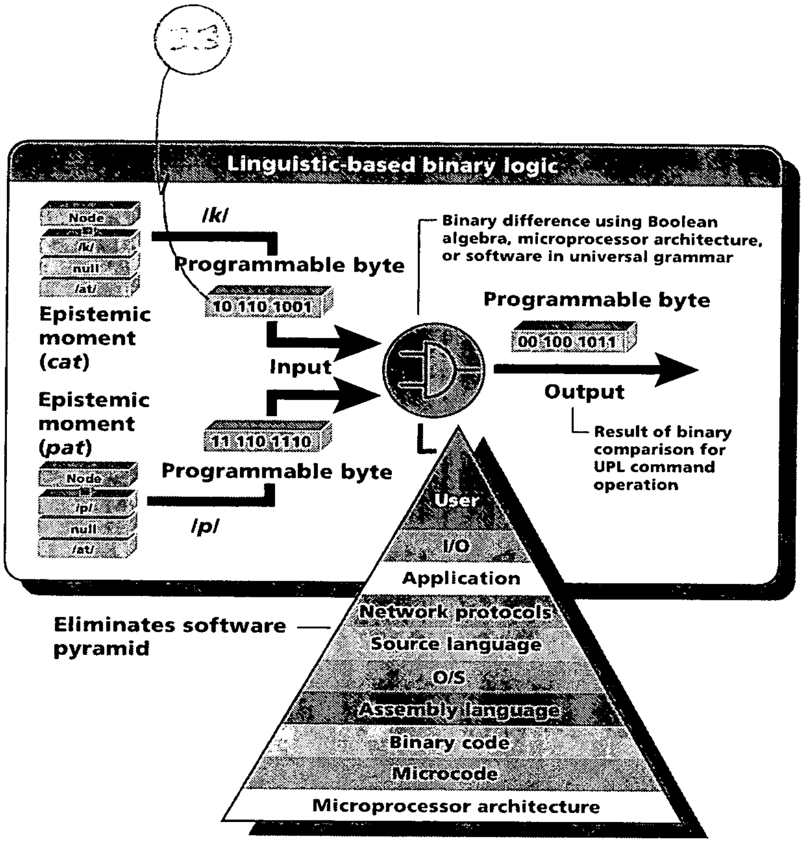

33
Linguistic-based binary logic
Node
/k/
null
/at/
Epistemic moment (*cat*)
/k/
Programmable byte
10 110 1001
Input
Binary difference using Boolean algebra, microprocessor architecture, or software in universal grammar
Programmable byte
00 100 1011
Output
Result of binary comparison for UPL command operation
Epistemic moment (*pat*)
11 110 1110
Programmable byte
Node
/p/
null
/at/
/p/
Eliminates software pyramid
User
I/O
Application
Network protocols
Source language
O/S
Assembly language
Binary code
Microcode
Microprocessor architecture

Fig.5

SHEET 6 OF 140

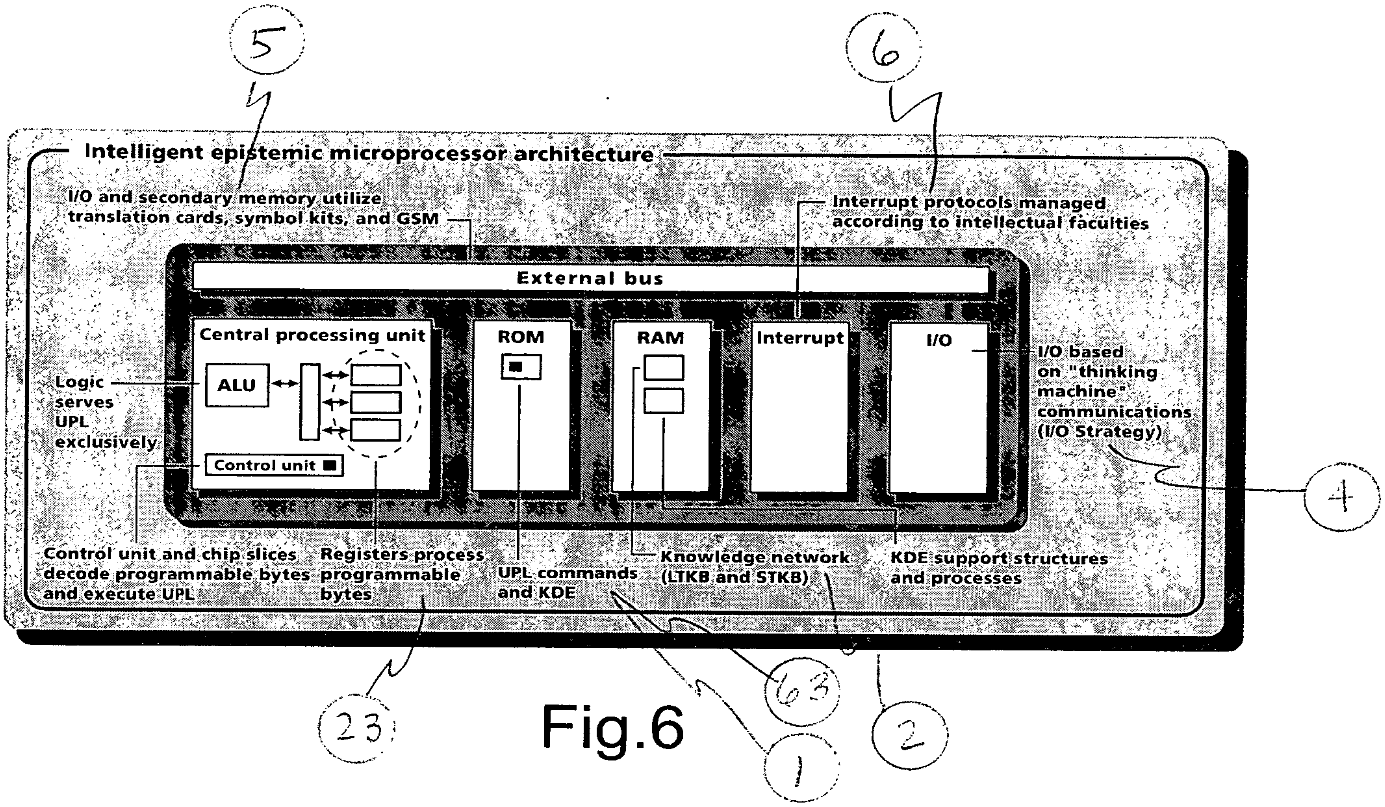
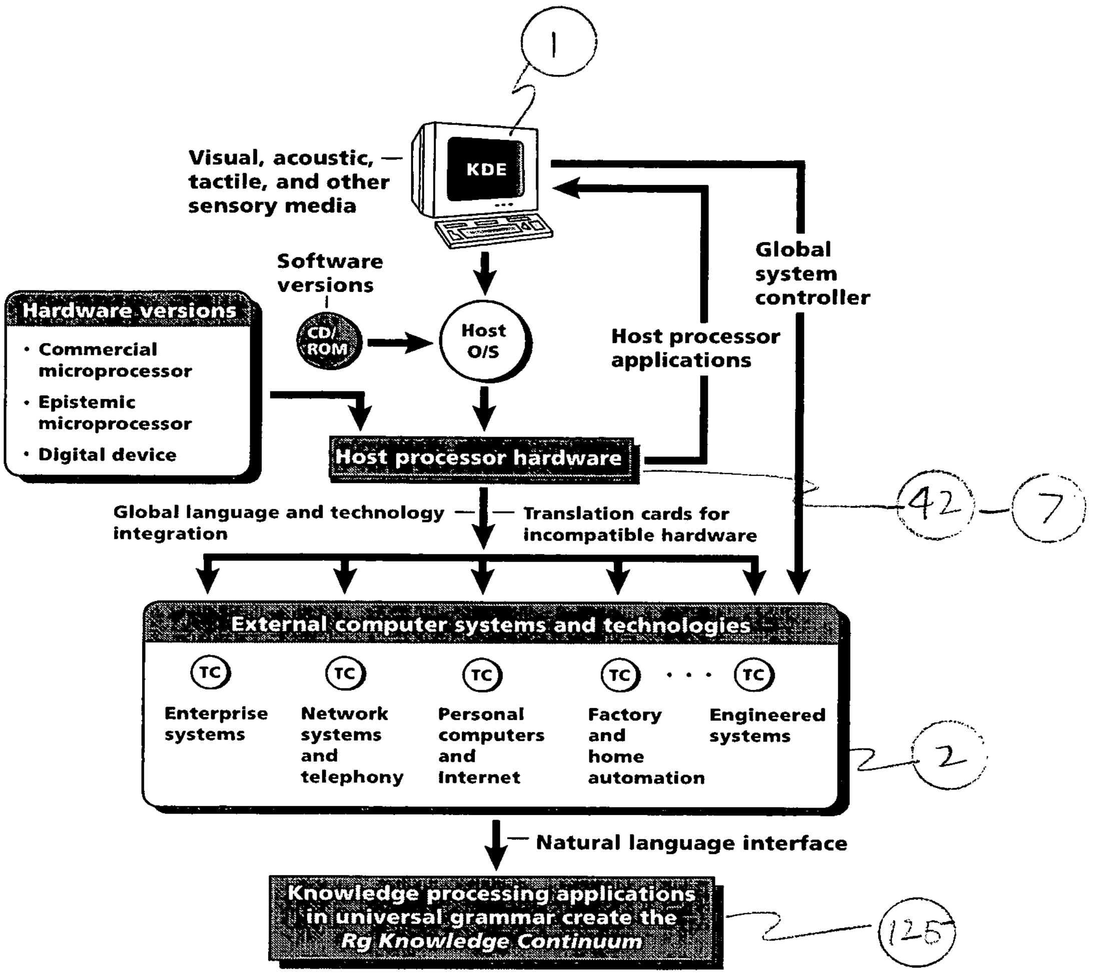

5
6
Intelligent epistemic microprocessor architecture
I/O and secondary memory utilize translation cards, symbol kits, and GSM
Interrupt protocols managed according to intellectual faculties
External bus
Central processing unit
ALU
Control unit
ROM
RAM
Interrupt
I/O
Logic serves UPL exclusively
I/O based on "thinking machine" communications (I/O Strategy)
4
Control unit and chip slices decode programmable bytes and execute UPL
Registers process programmable bytes
UPL commands and KDE
Knowledge network (LTKB and STKB)
KDE support structures and processes
23
63
1
2

Fig.6

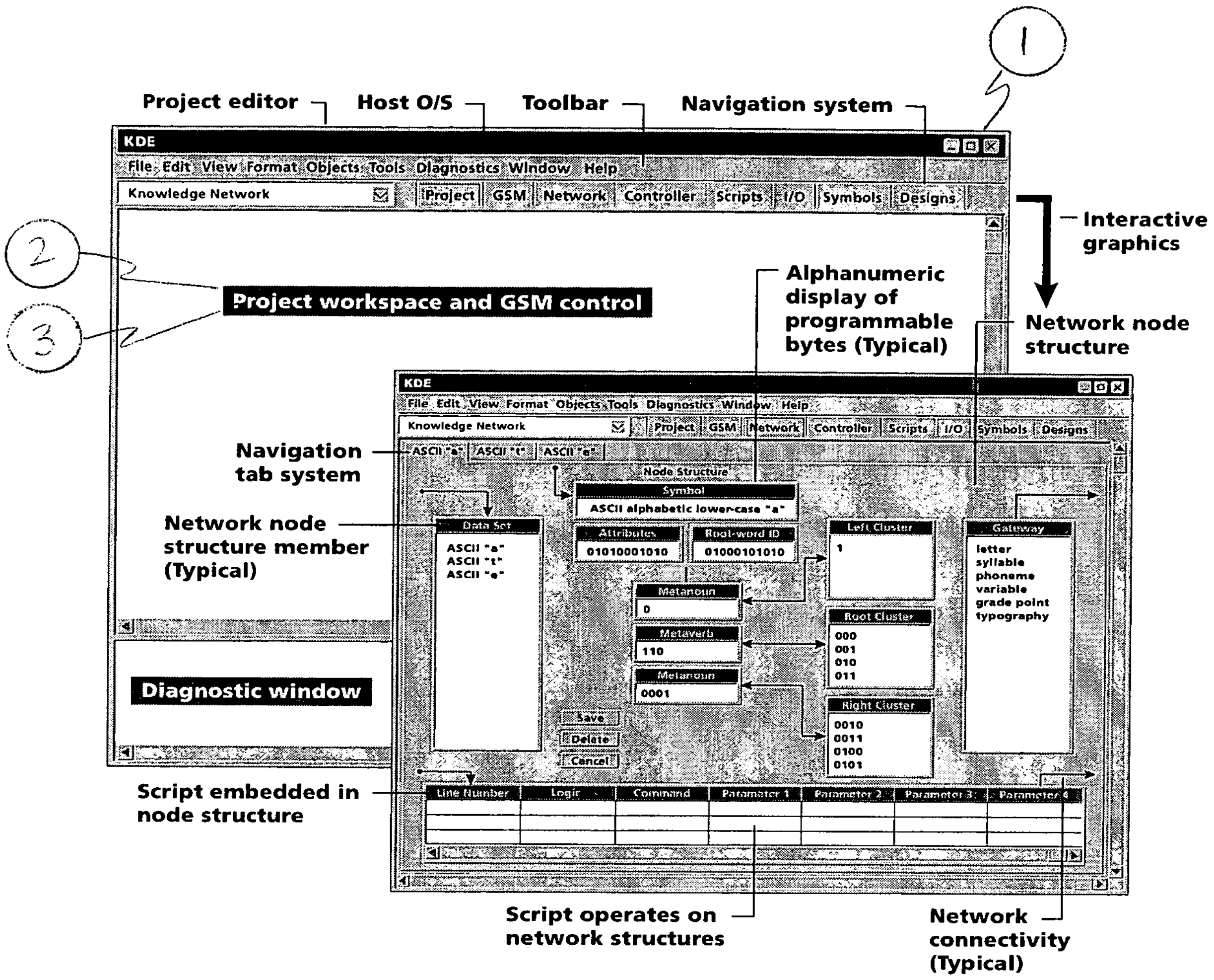
SHEET 7 OF 140



1
KDE
Visual, acoustic, tactile, and other sensory media
Software versions
CD/ ROM
Host O/S
Hardware versions
• Commercial microprocessor
• Epistemic microprocessor
• Digital device
Host processor hardware
Host processor applications
Global system controller
42
7
Global language and technology integration
Translation cards for incompatible hardware
External computer systems and technologies
TC
Enterprise systems
TC
Network systems and telephony
TC
Personal computers and Internet
TC
Factory and home automation
TC
Engineered systems
2
Natural language interface
Knowledge processing applications in universal grammar create the *Rg Knowledge Continuum*
125

Fig.7

SHEET 8 OF 140


1
2
3
Project editor
Host O/S
Toolbar
Navigation system
Interactive graphics
Alphanumeric display of programmable bytes (Typical)
Network node structure
Navigation tab system
Network node structure member (Typical)
Script embedded in node structure
Script operates on network structures
Network connectivity (Typical)
KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
Project workspace and GSM control
Diagnostic window
ASCII "a" ASCII "t" ASCII "e"
Node Structure
Save
Delete
Cancel

Symbol: ASCII alphabetic lower-case "a"

Data Set: ASCII "a" ASCII "t" ASCII "e"

Attributes: 01010001010

Root-word ID: 01000101010

Metanoun: 0

Metaverb: 110

Metanoun: 0001

Left Cluster: 1

Root Cluster: 000 001 010 011

Right Cluster: 0010 0011 0100 0101

Gateway: letter syllable phoneme variable grade point typography

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4

Fig.8

SHEET 9 OF 140

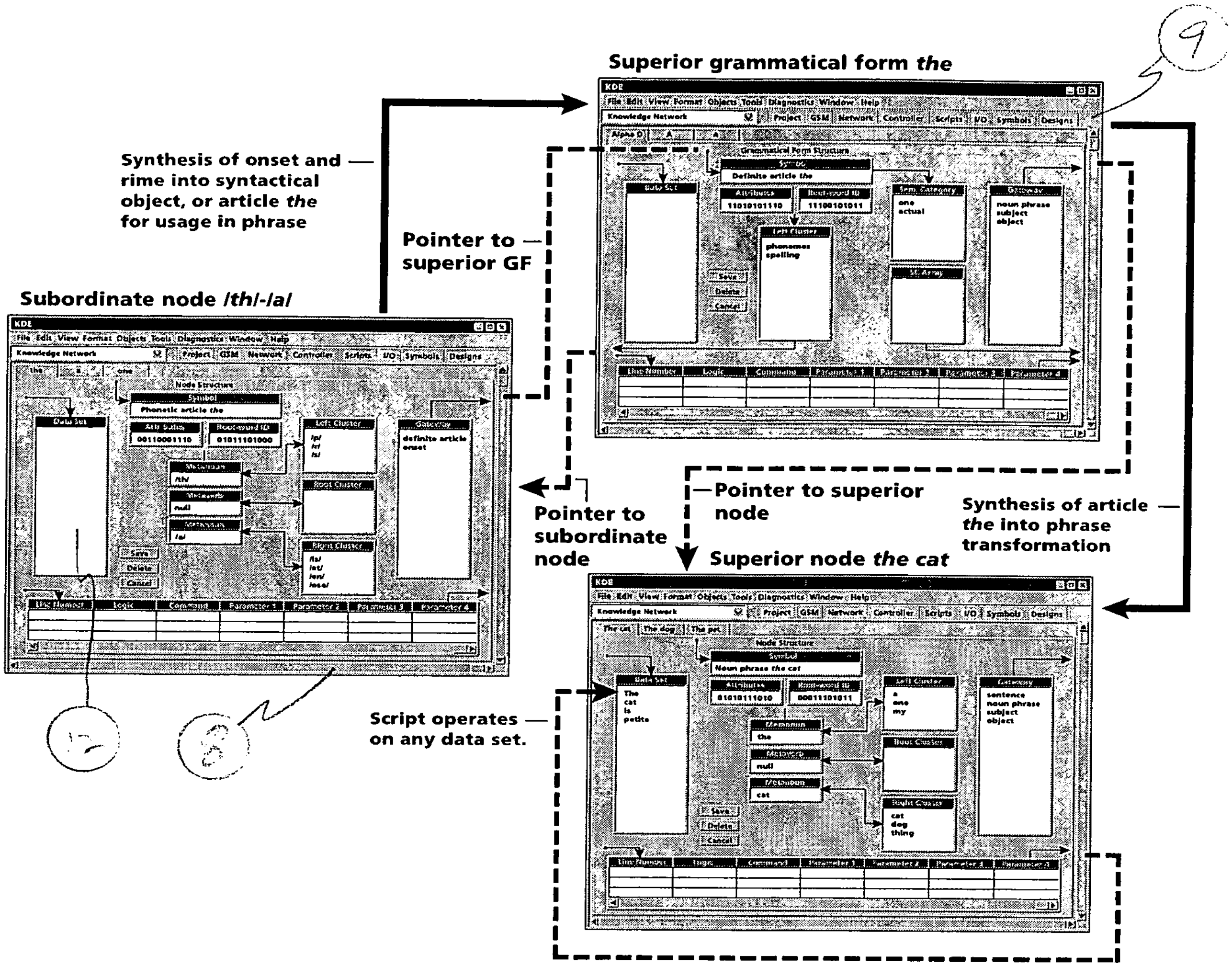
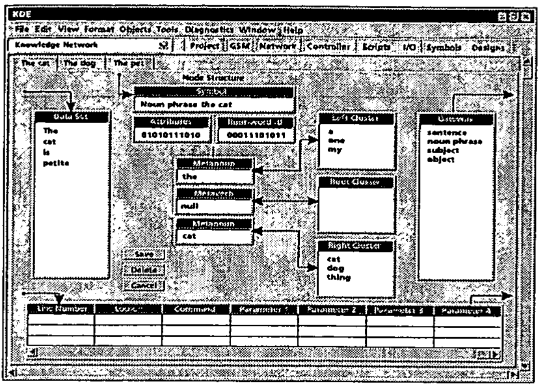


Fig.9

SHEET 10 OF 140

Node structure containing data set to be analyzed by sentence parser


KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
The cat
The dog
The pet
Node Structure
Save
Delete
Cancel

Symbol: Noun phrase the cat

Attributes: 01010111010

Root-word ID: 00011101011

Data Set: The cat is petite

Metanoun: the

Metaverb: null

Metanoun: cat

Left Cluster: a one my

Root Cluster:

Right Cluster: cat dog thing

Gateway: sentence noun phrase subject object

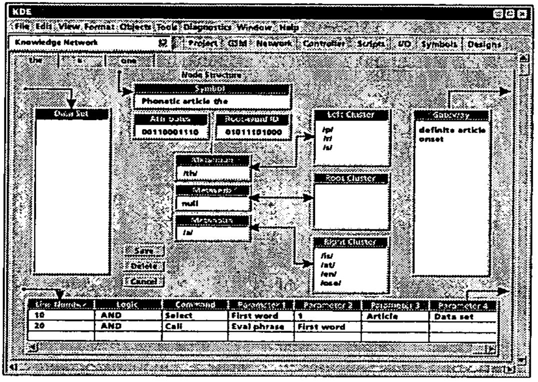
Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4

Scripts and data sets are embedded members of NL and GF structures

14

Node structure containing main sentence parser is invoked by superior calling function

Node structure containing main sentence parser


KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
the
a
one
Node Structure
Save
Delete
Cancel

Symbol: Phonetic article the

Attributes: 00110001110

Root-word ID: 01011101000

Data Set:

Metanoun: /th/

Metaverb: null

Metanoun: /a/

Left Cluster: /p/ /r/ /s/

Root Cluster:

Right Cluster: /is/ /at/ /en/ /ose/

Gateway: definite article onset

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Call	Eval phrase	First word		

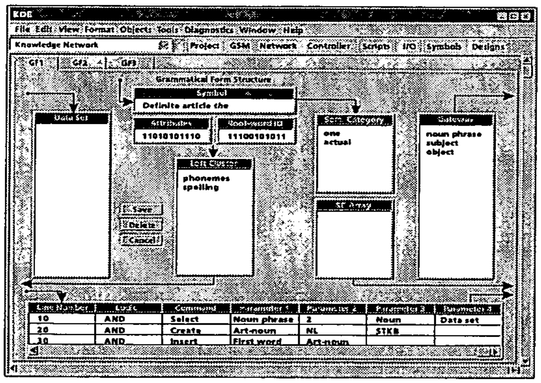
Main sentence parser analyzes data set by "selecting" first word and calling new script to evaluate noun phrase after determining the presence of the article *the.*

15

Invoked subordinate script analyzes noun phrase and returns control to main sentence parser.

Function call

GF structure containing subordinate noun-phrase parser


KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
GF1
GF2
GF3
Grammatical Form Structure
Save
Delete
Cancel

Symbol: Definite article the

Attributes: 11010101110

Root-word ID: 11100101011

Data Set:

Left Cluster: phonemes spelling

Sem. Category: one actual

SF Array:

Gateway: noun phrase subject object

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	Noun phrase	2	Noun	Data set
20	AND	Create	Art-noun	NL	STKB	
30	AND	Insert	First word	Art-noun		

Fig.10

SHEET 11 OF 140

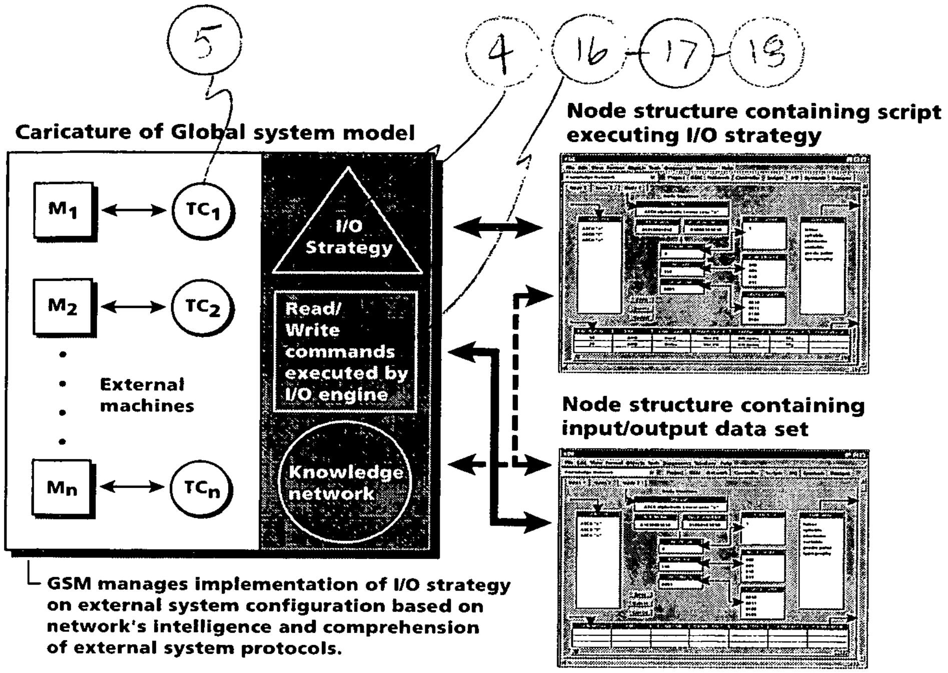
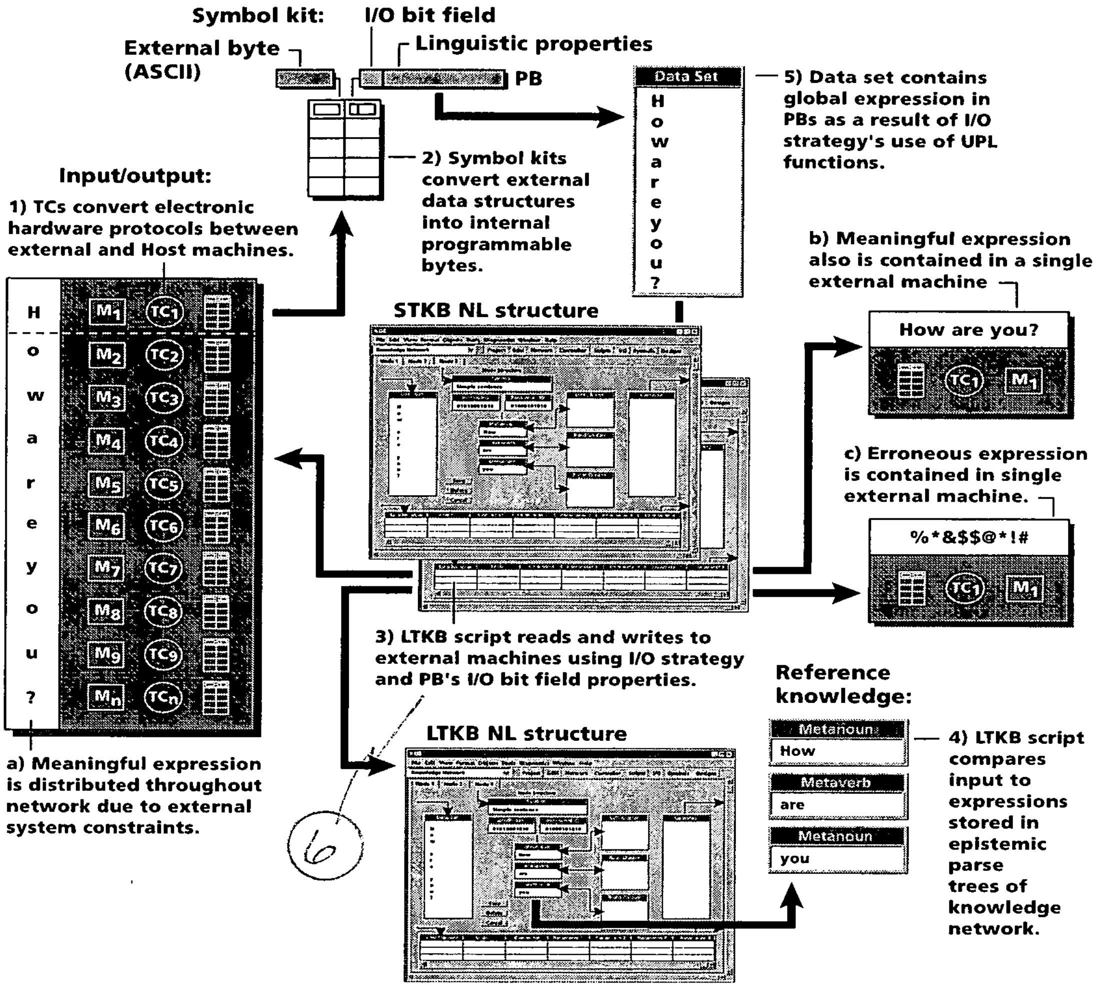

5
4
16
17
18
Caricature of Global system model
Node structure containing script executing I/O strategy
M1
TC1
I/O Strategy
M2
TC2
Read/ Write commands executed by I/O engine
External machines
Node structure containing input/output data set
Mn
TCn
Knowledge network
GSM manages implementation of I/O strategy on external system configuration based on network's intelligence and comprehension of external system protocols.

Fig.11



1) TCs convert electronic hardware protocols between external and Host machines.

2) Symbol kits convert external data structures into internal programmable bytes.

5) Data set contains global expression in PBs as a result of I/O strategy's use of UPL functions.

b) Meaningful expression also is contained in a single external machine

c) Erroneous expression is contained in single external machine.

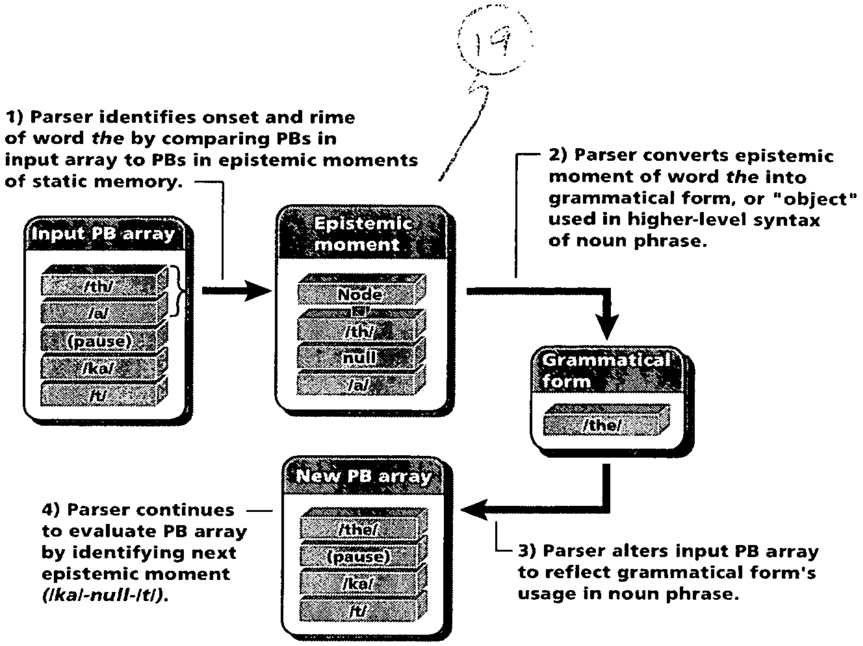
3) LTKB script reads and writes to external machines using I/O strategy and PB's I/O bit field properties.

a) Meaningful expression is distributed throughout network due to external system constraints.

4) LTKB script compares input to expressions stored in epistemic parse trees of knowledge network.

Fig.12

SHEET 13 OF 140



1) Parser identifies onset and rime of word *the* by comparing PBs in input array to PBs in epistemic moments of static memory.

2) Parser converts epistemic moment of word *the* into grammatical form, or "object" used in higher-level syntax of noun phrase.

3) Parser alters input PB array to reflect grammatical form's usage in noun phrase.

4) Parser continues to evaluate PB array by identifying next epistemic moment *(/ka/-null-/t/)*.

Fig.13

SHEET 14 OF 140

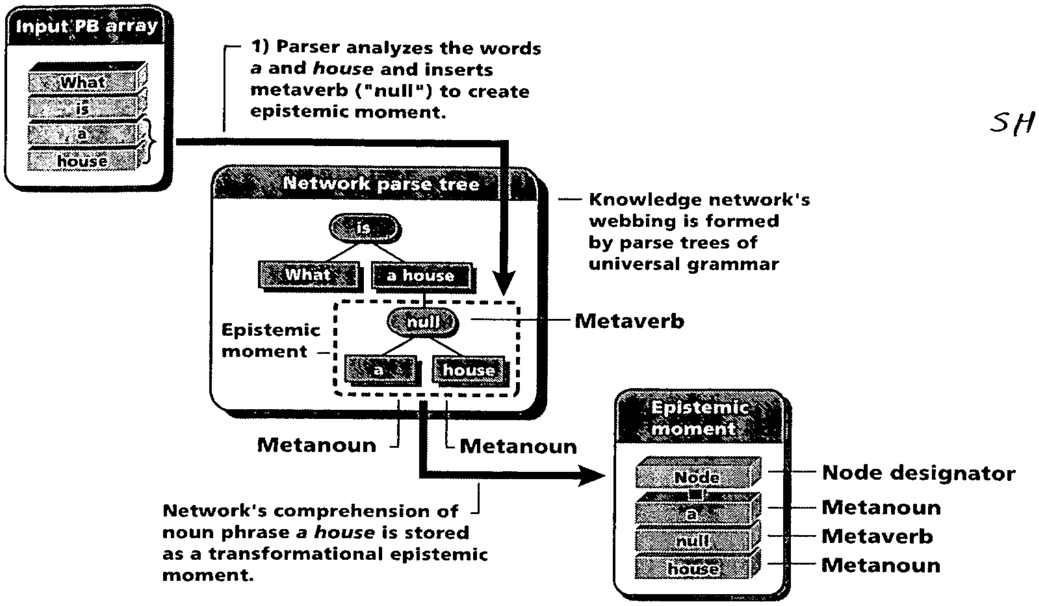

Input PB array
What
is
a
house
1) Parser analyzes the words *a* and *house* and inserts metaverb ("null") to create epistemic moment.
Network parse tree
is
What
a house
null
a
house
Knowledge network's webbing is formed by parse trees of universal grammar
Metaverb
Epistemic moment
Metanoun
Metanoun
Epistemic moment
Node
a
null
house
Node designator
Metanoun
Metaverb
Metanoun
Network's comprehension of noun phrase *a house* is stored as a transformational epistemic moment.

Fig.14

SHEET 15 OF 140

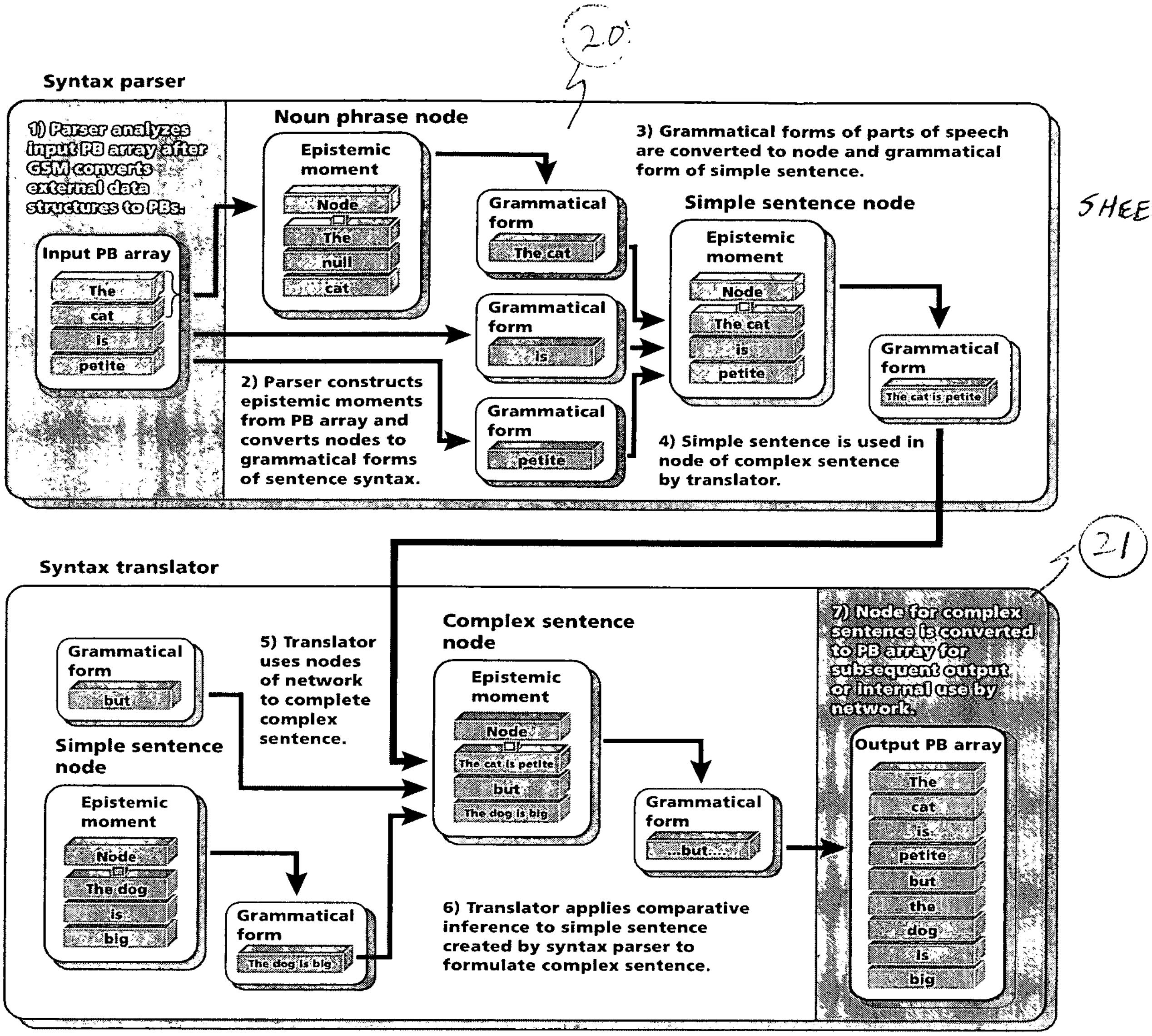
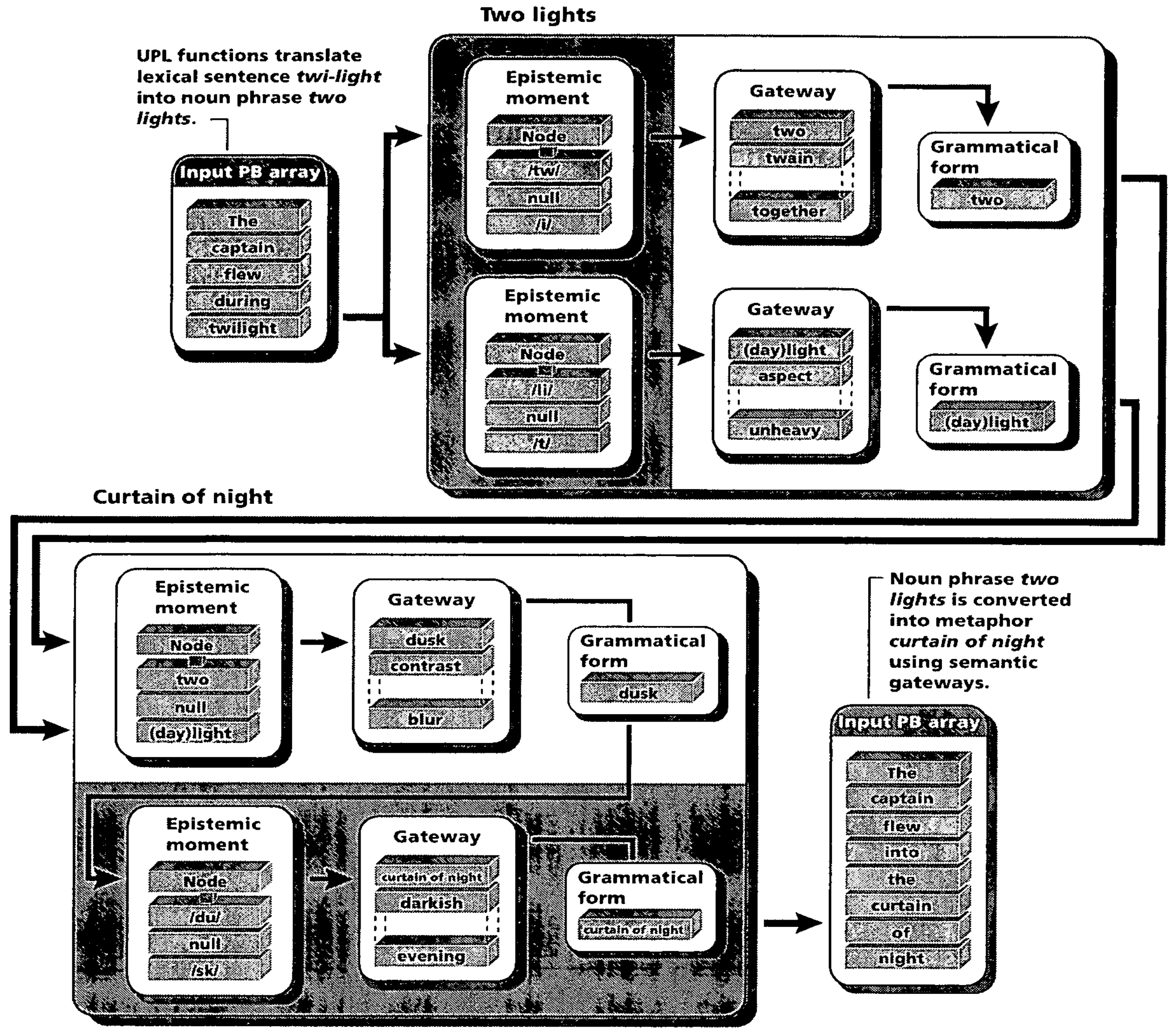

20
21
Syntax parser
1) Parser analyzes input PB array after GSM converts external data structures to PBs.
Input PB array
The
cat
is
petite
Noun phrase node
Epistemic moment
Node
The
null
cat
2) Parser constructs epistemic moments from PB array and converts nodes to grammatical forms of sentence syntax.
Grammatical form
The cat
Grammatical form
is
Grammatical form
petite
3) Grammatical forms of parts of speech are converted to node and grammatical form of simple sentence.
Simple sentence node
Epistemic moment
Node
The cat
is
petite
Grammatical form
The cat is petite
4) Simple sentence is used in node of complex sentence by translator.
Syntax translator
Grammatical form
but
5) Translator uses nodes of network to complete complex sentence.
Simple sentence node
Epistemic moment
Node
The dog
is
big
Grammatical form
The dog is big
Complex sentence node
Epistemic moment
Node
The cat is petite
but
The dog is big
Grammatical form
...but...
6) Translator applies comparative inference to simple sentence created by syntax parser to formulate complex sentence.
7) Node for complex sentence is converted to PB array for subsequent output or internal use by network.
Output PB array
The
cat
is
petite
but
the
dog
is
big

Fig.15



UPL functions translate lexical sentence *twi-light* into noun phrase *two lights*.

Noun phrase *two lights* is converted into metaphor *curtain of night* using semantic gateways.

Fig.16(a)

SHEET 17 OF 140

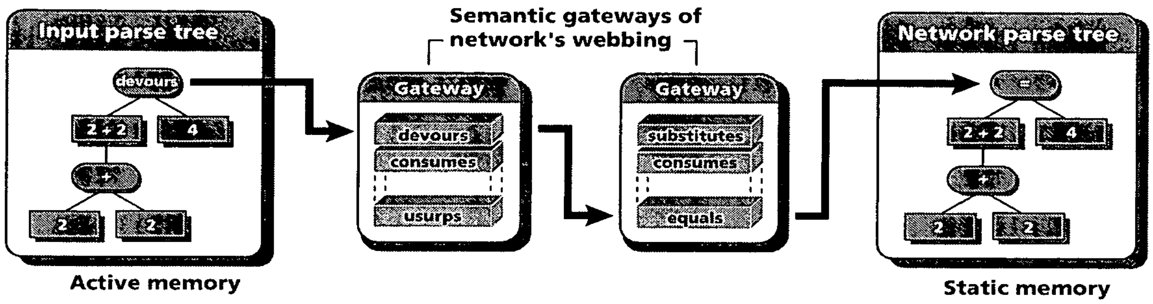
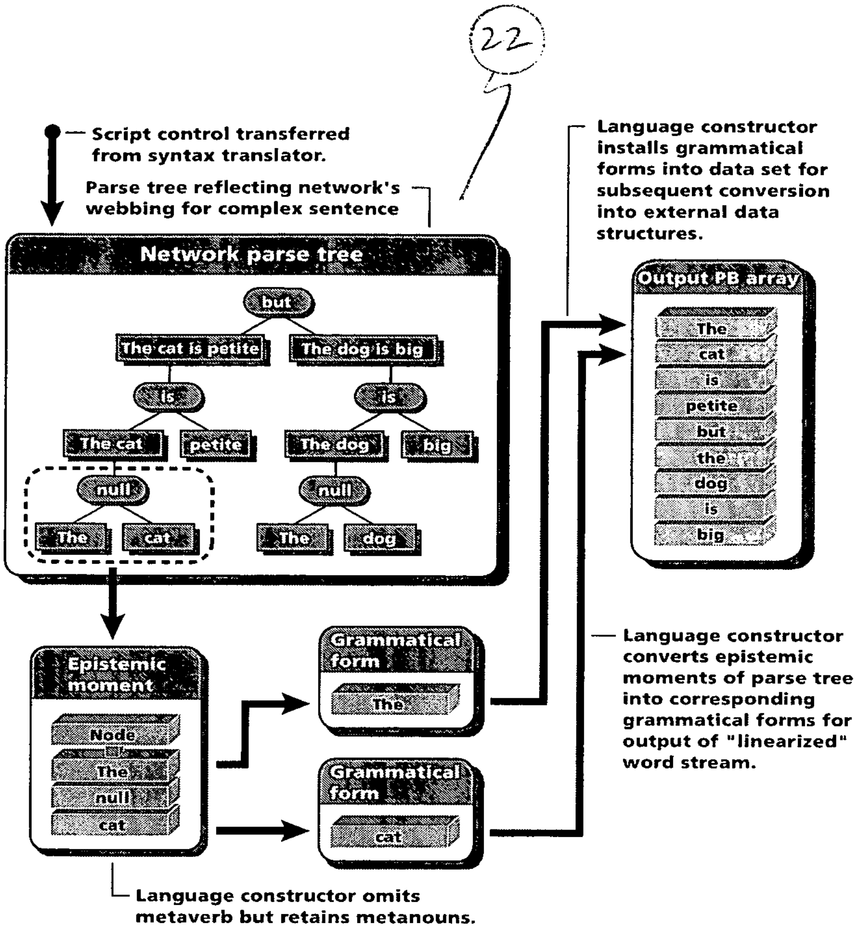


Fig.16(b)



Script control transferred from syntax translator.

Parse tree reflecting network's webbing for complex sentence

Language constructor installs grammatical forms into data set for subsequent conversion into external data structures.

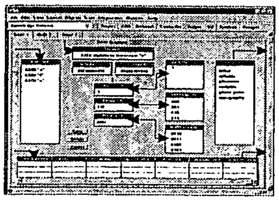
Language constructor converts epistemic moments of parse tree into corresponding grammatical forms for output of "linearized" word stream.

Language constructor omits metaverb but retains metanouns.

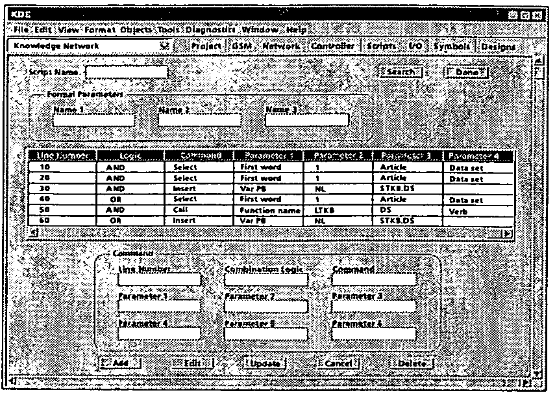
Fig.17

1) Developer creates scripts that are embedded in network and operate on structure members.

Node structure containing script



Script editor



Script Name

Formal Parameters

Name 1 / Name 2 / Name 3

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Command

Line Number / Combination Logic / Command

Parameter 1 / Parameter 2 / Parameter 3

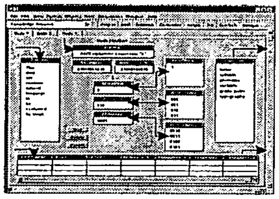
Parameter 4 / Parameter 5 / Parameter 6

b) Function "call" specifies network member to be operated on.

Data Set

This data set contains natural language to be evaluated by script.

Node structure containing data set operated on by script



c) Pointer indicates where previous function disengaged to invoke new function.

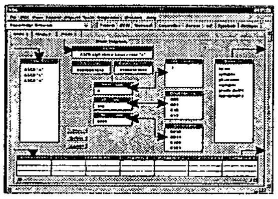
AND/OR Logic

Task A
AND
Task B
AND
Task C
OR
Task D
AND
Task E
OR
Task F

Function call

Parameter 1	Parameter 2	Parameter 3	Parameter 4
LTKB/STKB	STRUCTURE MEMBER	POINTER	DEFAULT ON/OFF

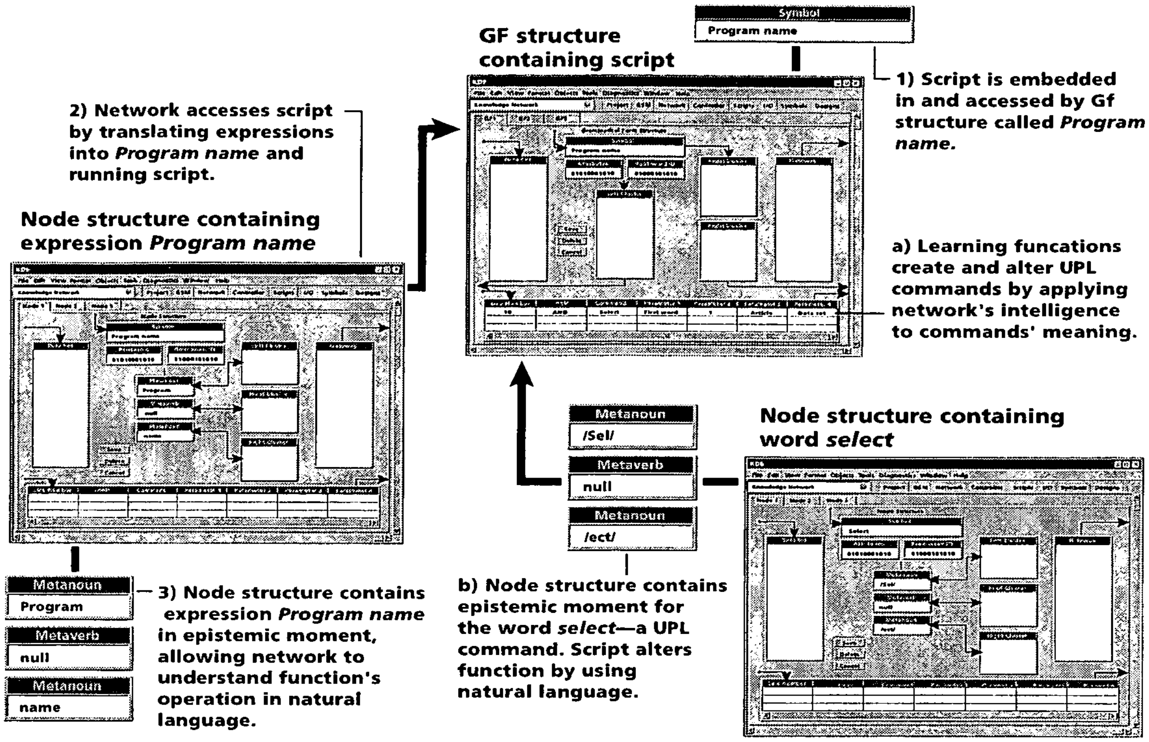
Node structure containing invoked function



a) Function call allows invoked function to continue parsing and translating logic through specification of the knowledge base (LTKB/STKB), Node or GF structure member, command pointers, and default status.

2) UPL function operates by evaluating linguistic conditions and taking related actions. If condition fails, function proceeds to next "OR" command. Final "OR" command is default action.

Fig.18



GF structure containing script

1) Script is embedded in and accessed by Gf structure called *Program name.*

2) Network accesses script by translating expressions into *Program name* and running script.

Node structure containing expression *Program name*

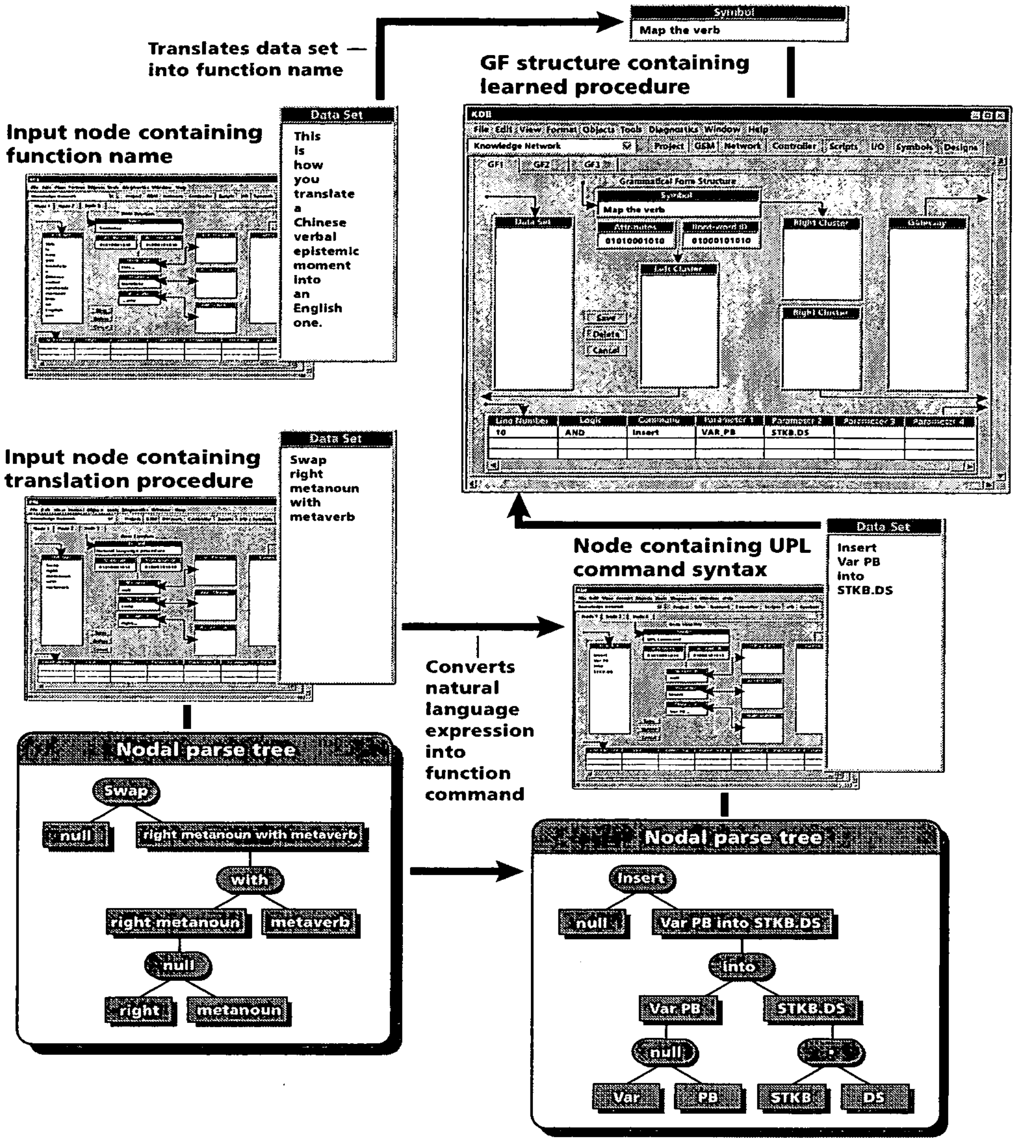
a) Learning funcations create and alter UPL commands by applying network's intelligence to commands' meaning.

Node structure containing word *select*

b) Node structure contains epistemic moment for the word *select*—a UPL command. Script alters function by using natural language.

3) Node structure contains expression *Program name* in epistemic moment, allowing network to understand function's operation in natural language.

Fig.19


Translates data set into function name
Symbol
Map the verb
GF structure containing learned procedure
Input node containing function name
Data Set
KDB
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
GF1 GF2 GF3
Grammatical Form Structure
Symbol
Map the verb
Attributes
01010001010
Next-word ID
01000101010
Data Set
Left Cluster
Right Cluster
Right Cluster
Gateway
Save
Delete
Cancel
Input node containing translation procedure
Data Set
Node containing UPL command syntax
Data Set
Converts natural language expression into function command
Nodal parse tree
Swap
null
right metanoun with metaverb
with
right metanoun
metaverb
null
right
metanoun
Nodal parse tree
Insert
null
Var PB into STKB.DS
into
Var.PB
STKB.DS
null
.
Var
PB
STKB
DS

This is how you translate a Chinese verbal epistemic moment into an English one.

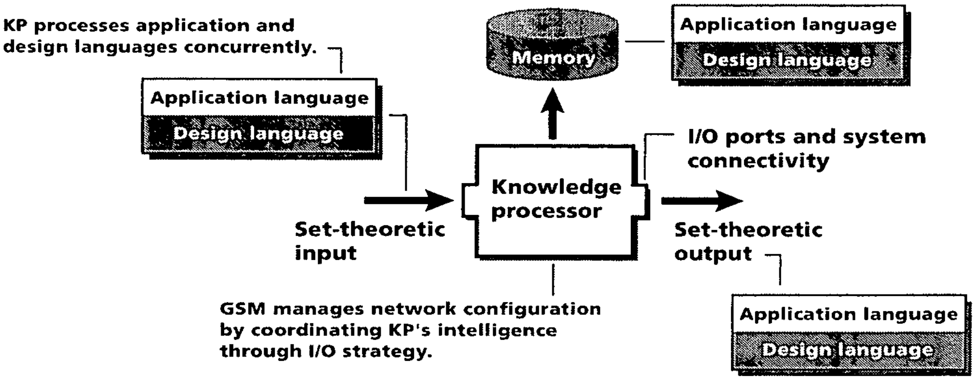
Swap right metanoun with metaverb

Insert Var PB into STKB.DS

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Insert	VAR.PB	STKB.DS		

Fig. 20

SHEET 22 OF 140



KP processes application and design languages concurrently.
Application language
Design language
Memory
Application language
Design language
I/O ports and system connectivity
Knowledge processor
Set-theoretic input
Set-theoretic output
GSM manages network configuration by coordinating KP's intelligence through I/O strategy.
Application language
Design language

Fig. 21

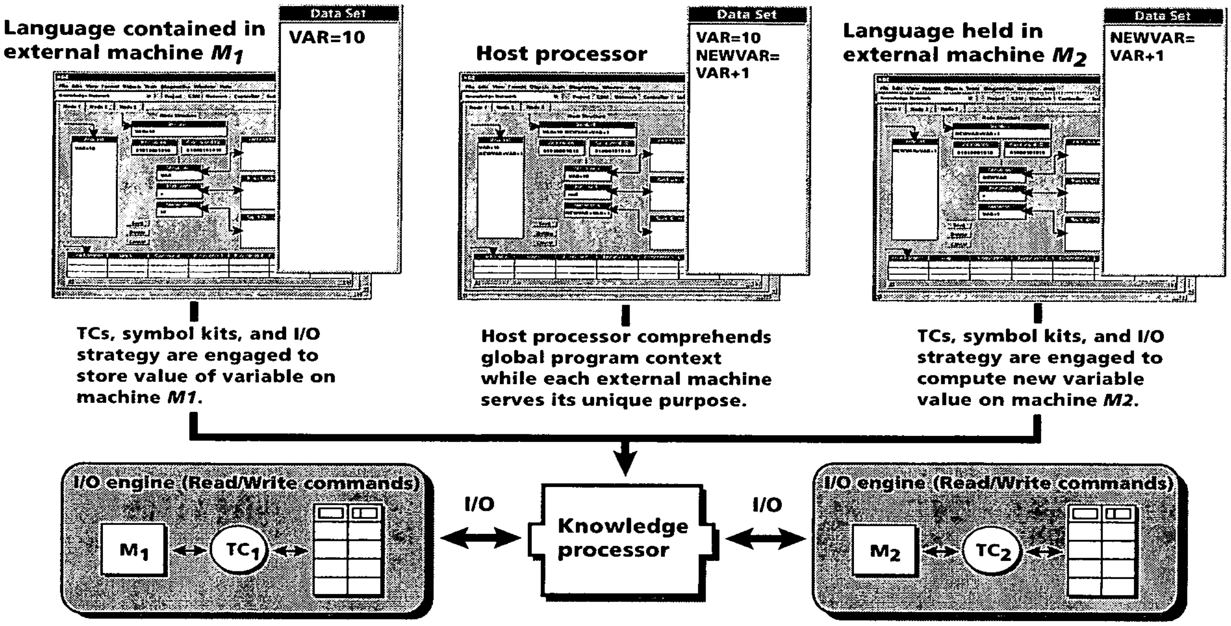

Language contained in external machine M_1
Data Set
VAR=10
Host processor
Data Set
VAR=10
NEWVAR=
VAR+1
Language held in external machine M_2
Data Set
NEWVAR=
VAR+1
TCs, symbol kits, and I/O strategy are engaged to store value of variable on machine *M1*.
Host processor comprehends global program context while each external machine serves its unique purpose.
TCs, symbol kits, and I/O strategy are engaged to compute new variable value on machine *M2*.
I/O engine (Read/Write commands)
M_1
TC_1
I/O
Knowledge processor
I/O
I/O engine (Read/Write commands)
M_2
TC_2

Fig. 22

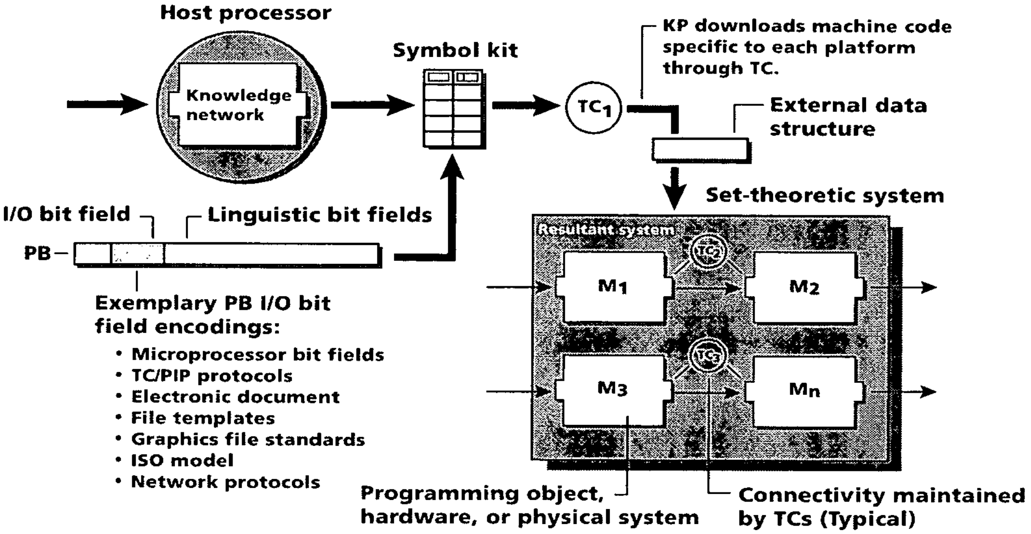

Host processor
Knowledge network
Symbol kit
TC1
KP downloads machine code specific to each platform through TC.
External data structure
Set-theoretic system
I/O bit field
Linguistic bit fields
PB
Exemplary PB I/O bit field encodings:
• Microprocessor bit fields
• TC/PIP protocols
• Electronic document
• File templates
• Graphics file standards
• ISO model
• Network protocols
Resultant system
TC2
TC3
M1
M2
M3
Mn
Programming object, hardware, or physical system
Connectivity maintained by TCs (Typical)

Fig. 23

SHEET 25 OF 140

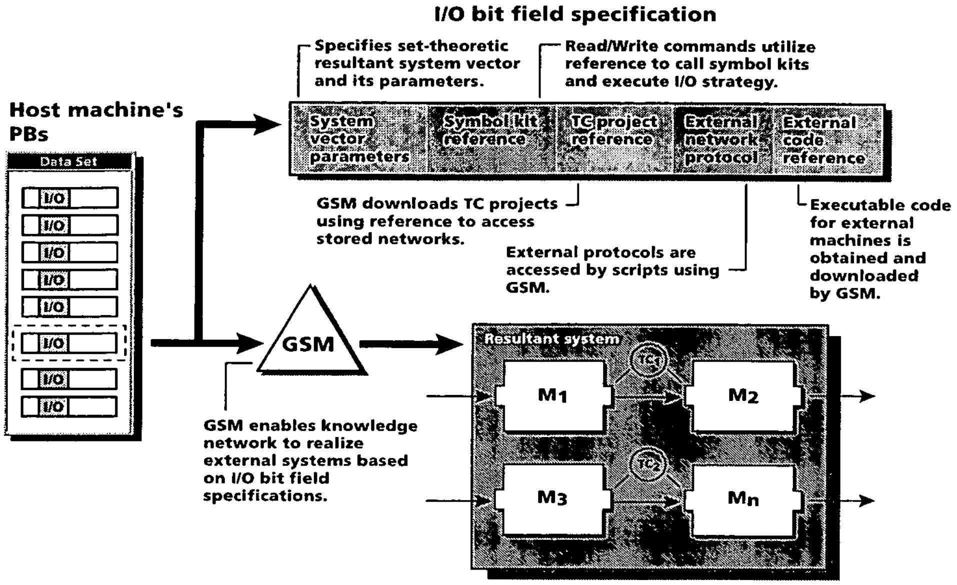

I/O bit field specification
Specifies set-theoretic resultant system vector and its parameters.
Read/Write commands utilize reference to call symbol kits and execute I/O strategy.
Host machine's PBs
Data Set
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
System vector parameters
Symbol kit reference
TC project reference
External network protocol
External code reference
GSM downloads TC projects using reference to access stored networks.
External protocols are accessed by scripts using GSM.
Executable code for external machines is obtained and downloaded by GSM.
GSM
GSM enables knowledge network to realize external systems based on I/O bit field specifications.
Resultant system
TC1
TC2
M1
M2
M3
Mn

Fig. 24

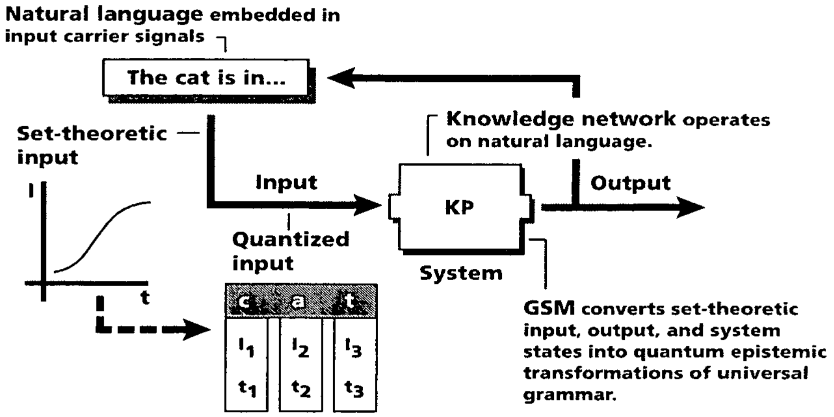

Natural language embedded in input carrier signals
The cat is in...
Set-theoretic input
Knowledge network operates on natural language.
I
t
Input
KP
Output
Quantized input
System
c
a
t
I_1
t_1
I_2
t_2
I_3
t_3
GSM converts set-theoretic input, output, and system states into quantum epistemic transformations of universal grammar.

Fig. 25

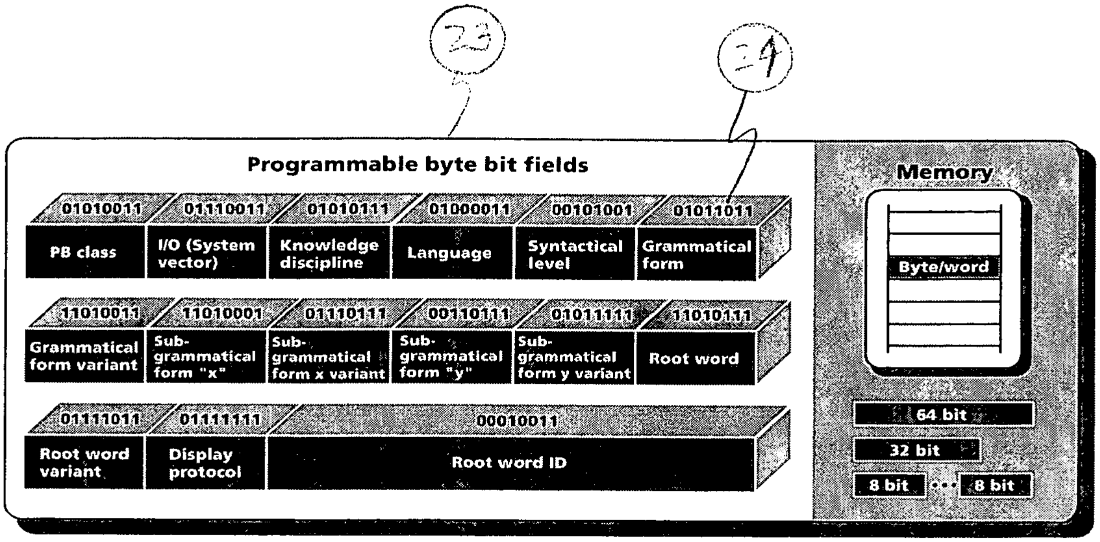

23
24
Programmable byte bit fields
01010011
PB class
01110011
I/O (System vector)
01010111
Knowledge discipline
01000011
Language
00101001
Syntactical level
01011011
Grammatical form
11010011
Grammatical form variant
11010001
Sub-grammatical form "x"
01110111
Sub-grammatical form x variant
00110111
Sub-grammatical form "y"
01011111
Sub-grammatical form y variant
11010111
Root word
01111011
Root word variant
01111111
Display protocol
00010011
Root word ID
Memory
Byte/word
64 bit
32 bit
8 bit
8 bit

Fig. 26

SHEET 28 OF 140

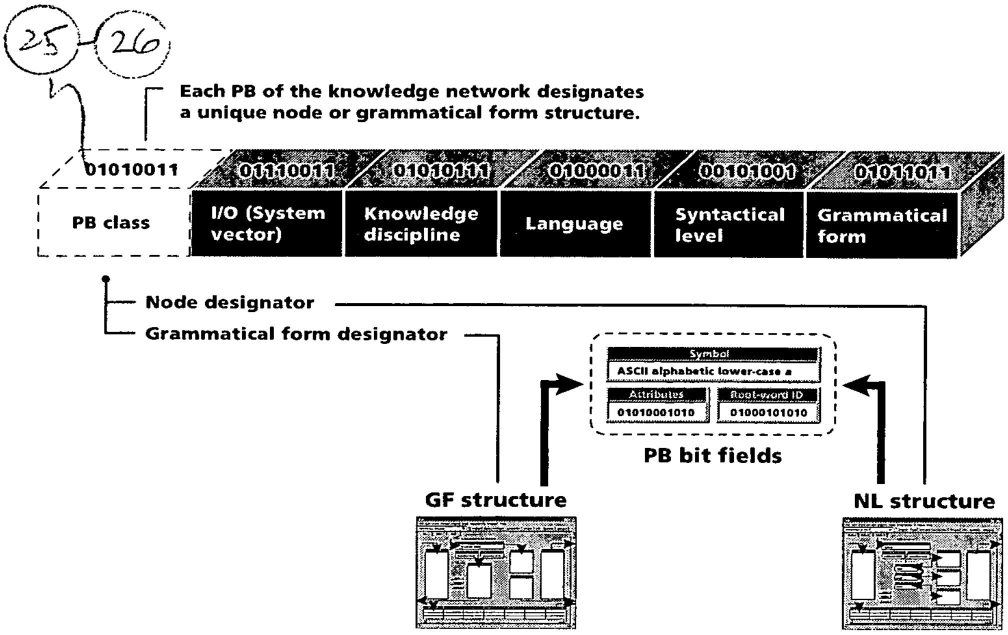


Fig. 27

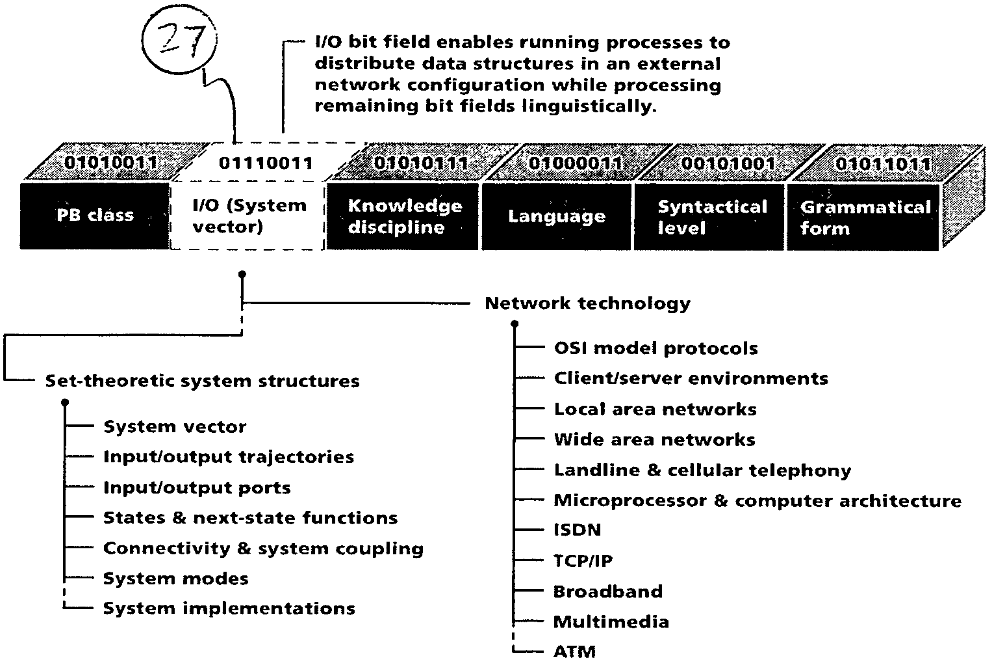

27
I/O bit field enables running processes to distribute data structures in an external network configuration while processing remaining bit fields linguistically.
01010011
01110011
01010111
01000011
00101001
01011011
PB class
I/O (System vector)
Knowledge discipline
Language
Syntactical level
Grammatical form
Network technology
OSI model protocols
Client/server environments
Local area networks
Wide area networks
Landline & cellular telephony
Microprocessor & computer architecture
ISDN
TCP/IP
Broadband
Multimedia
ATM
Set-theoretic system structures
System vector
Input/output trajectories
Input/output ports
States & next-state functions
Connectivity & system coupling
System modes
System implementations

Fig. 28

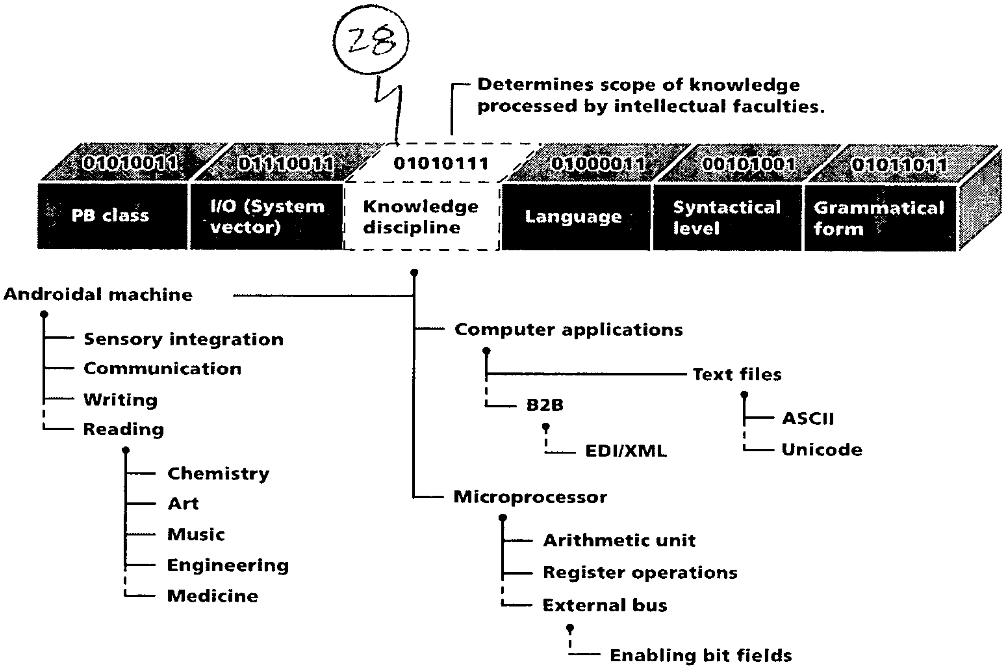

28
Determines scope of knowledge processed by intellectual faculties.
01010011
PB class
01110011
I/O (System vector)
01010111
Knowledge discipline
01000011
Language
00101001
Syntactical level
01011011
Grammatical form
Androidal machine
Sensory integration
Communication
Writing
Reading
Chemistry
Art
Music
Engineering
Medicine
Computer applications
Text files
B2B
EDI/XML
ASCII
Unicode
Microprocessor
Arithmetic unit
Register operations
External bus
Enabling bit fields

Fig. 29

SHEET 31 OF 140

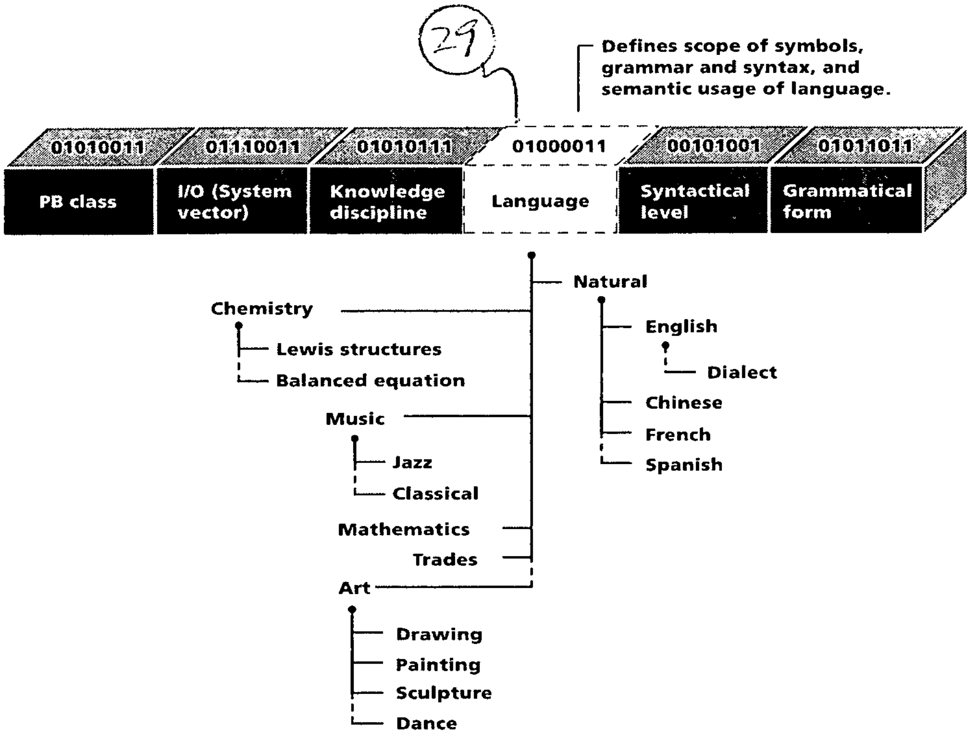


Fig. 30

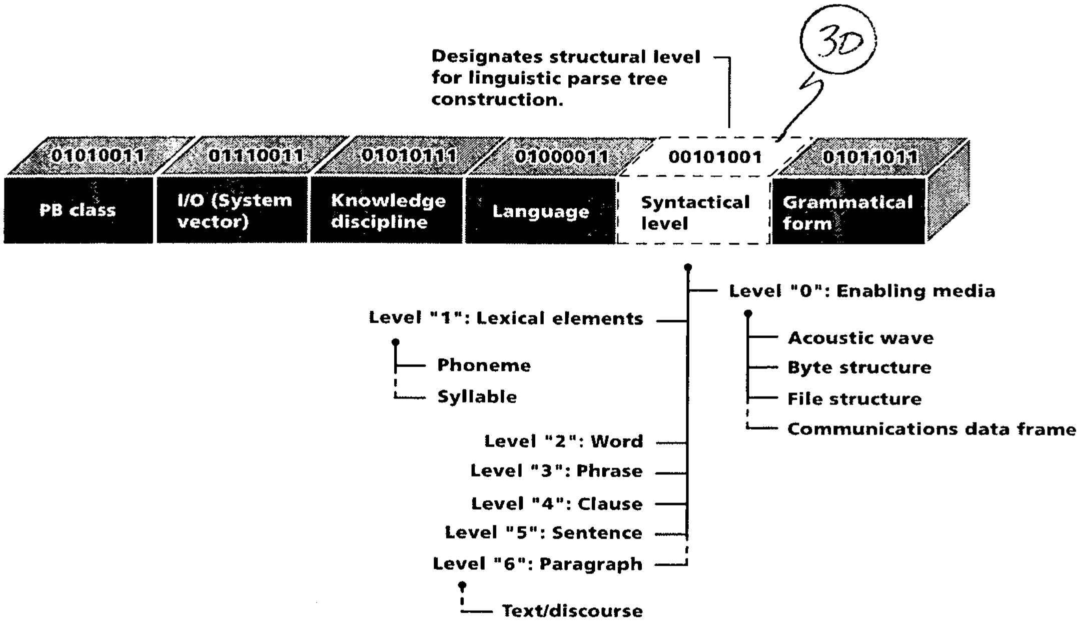

30
Designates structural level for linguistic parse tree construction.
01010011
01110011
01010111
01000011
00101001
01011011
PB class
I/O (System vector)
Knowledge discipline
Language
Syntactical level
Grammatical form
Level "0": Enabling media
Acoustic wave
Byte structure
File structure
Communications data frame
Level "1": Lexical elements
Phoneme
Syllable
Level "2": Word
Level "3": Phrase
Level "4": Clause
Level "5": Sentence
Level "6": Paragraph
Text/discourse

Fig. 31

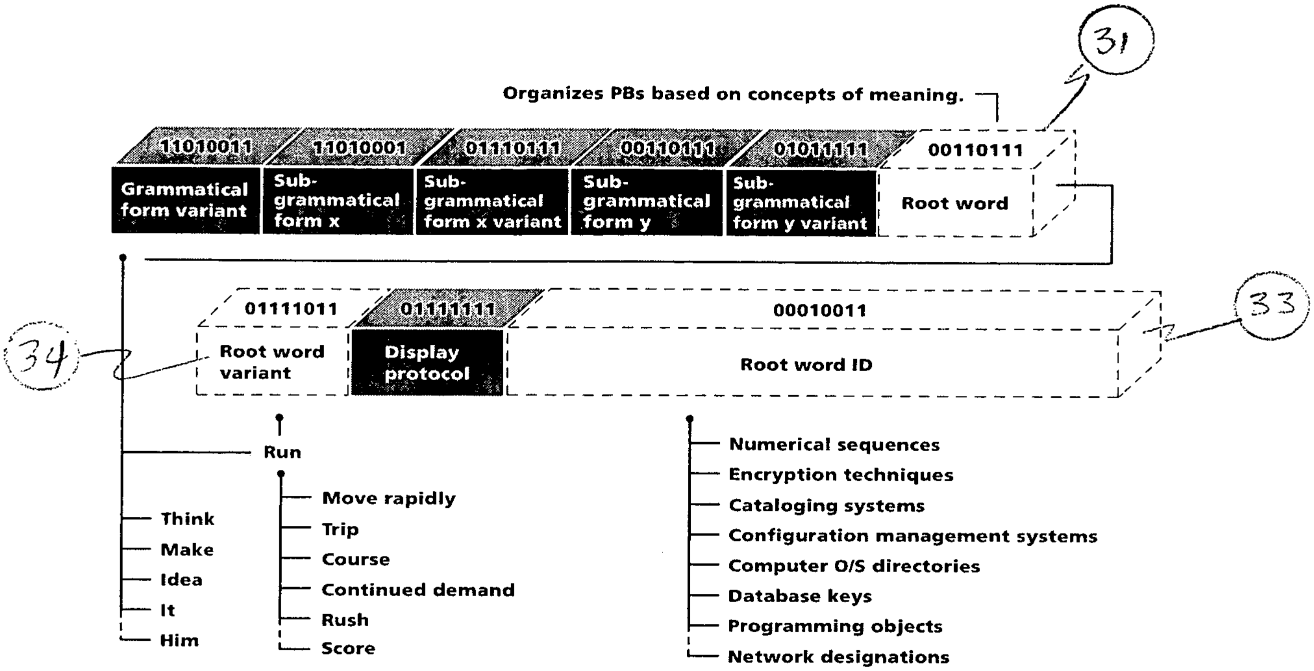

31
Organizes PBs based on concepts of meaning.
11010011
11010001
01110111
00110111
01011111
00110111
Grammatical form variant
Sub-grammatical form x
Sub-grammatical form x variant
Sub-grammatical form y
Sub-grammatical form y variant
Root word
01111011
01111111
00010011
33
34
Root word variant
Display protocol
Root word ID
Run
Move rapidly
Trip
Course
Continued demand
Rush
Score
Think
Make
Idea
It
Him
Numerical sequences
Encryption techniques
Cataloging systems
Configuration management systems
Computer O/S directories
Database keys
Programming objects
Network designations

Fig. 32

SHEET 34 OF 140

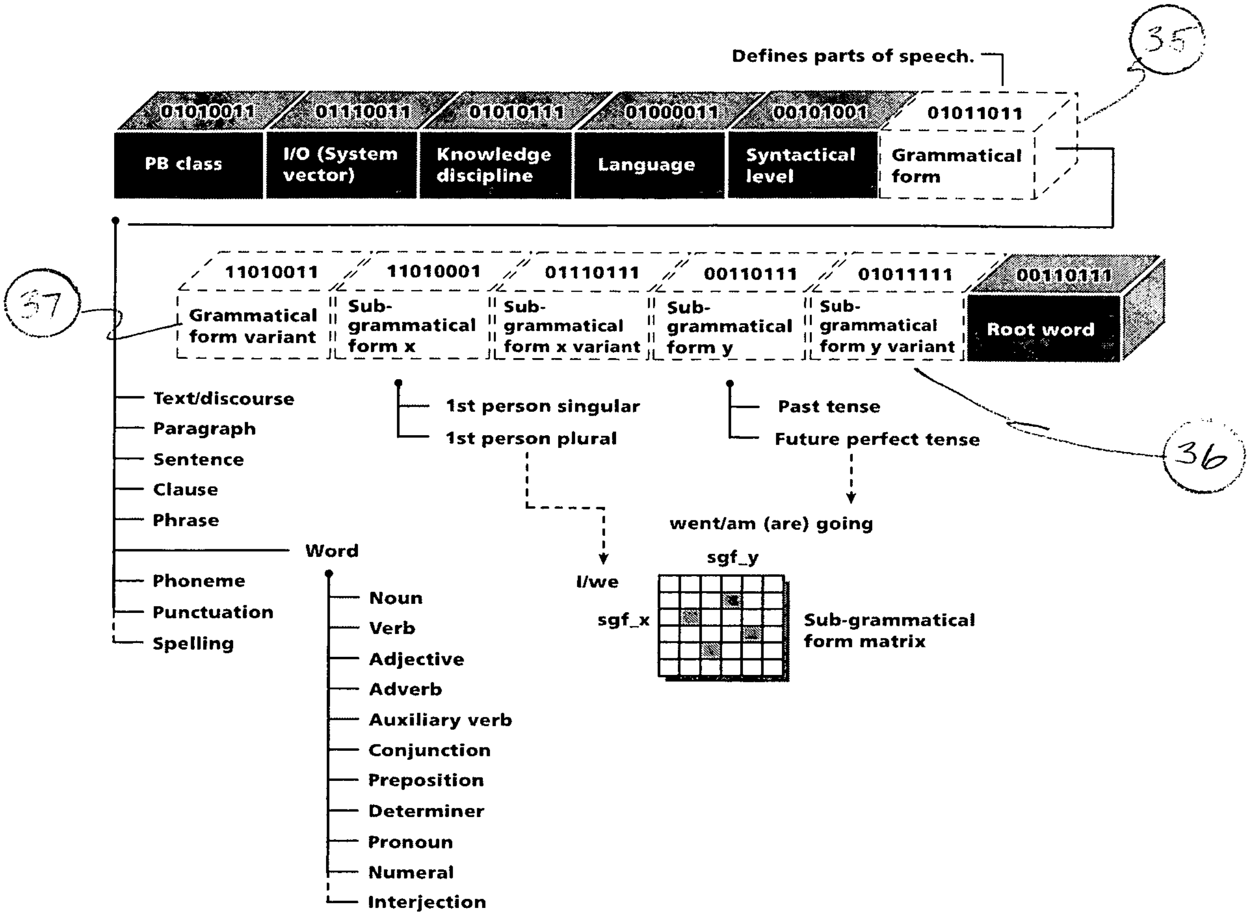
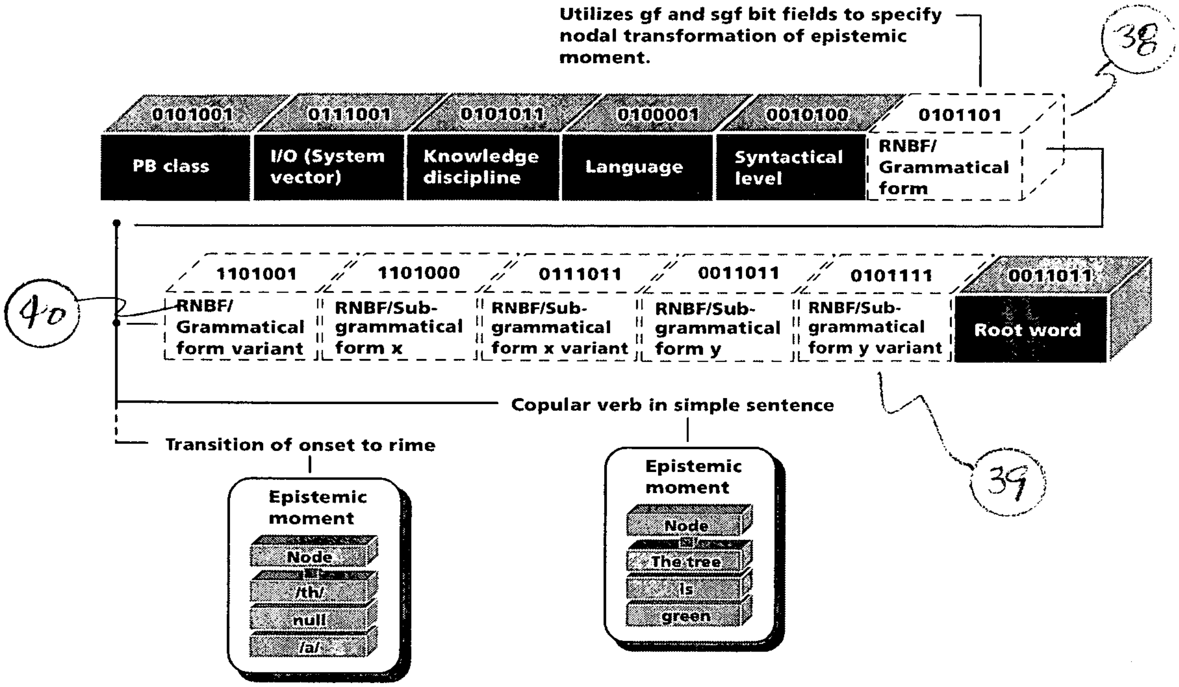


Fig. 33

SHEET 35 OF 110



Utilizes gf and sgf bit fields to specify nodal transformation of epistemic moment.
38
0101001
PB class
0111001
I/O (System vector)
0101011
Knowledge discipline
0100001
Language
0010100
Syntactical level
0101101
RNBF/ Grammatical form
40
1101001
RNBF/ Grammatical form variant
1101000
RNBF/Sub-grammatical form x
0111011
RNBF/Sub-grammatical form x variant
0011011
RNBF/Sub-grammatical form y
0101111
RNBF/Sub-grammatical form y variant
0011011
Root word
Copular verb in simple sentence
Transition of onset to rime
39
Epistemic moment
Node
/th/
null
/a/
Epistemic moment
Node
The tree
is
green

Fig. 34

SHEET 36 OF 140

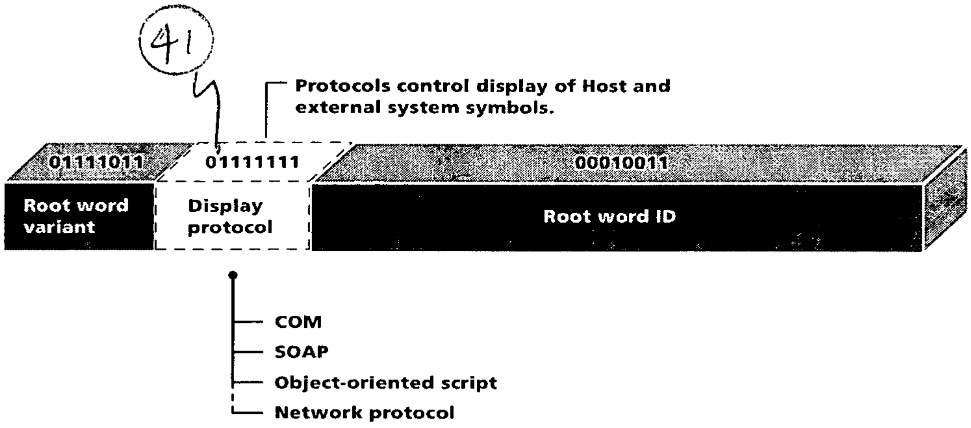
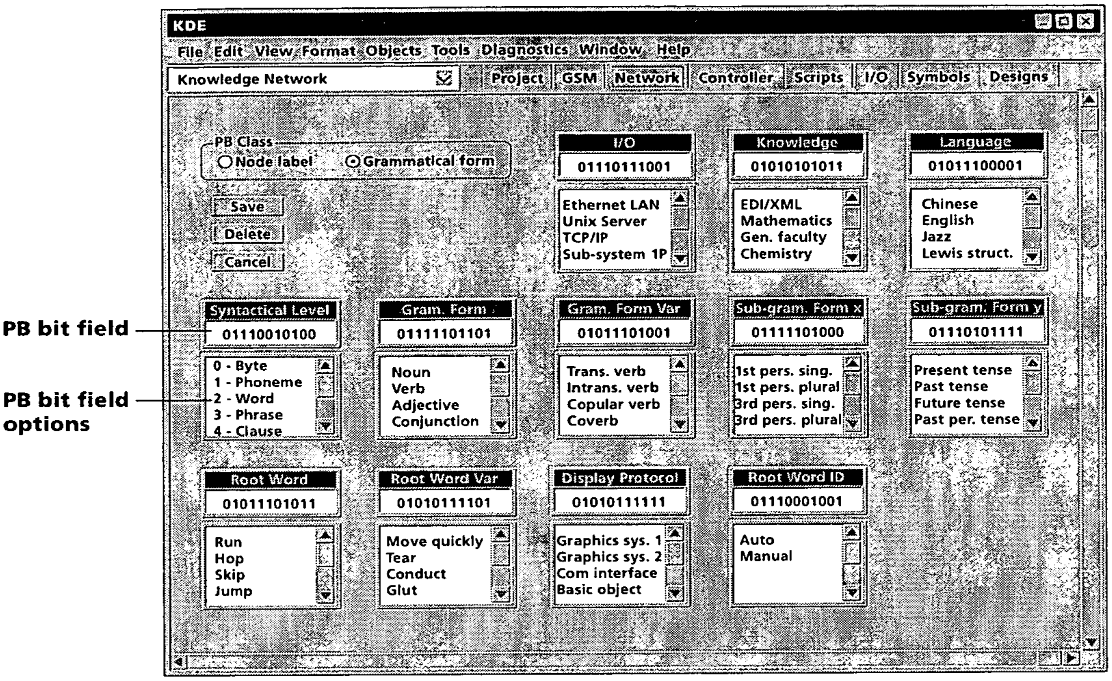


Fig. 35

SHEET 37 OF 140


KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
Save
Delete
Cancel
PB bit field
PB bit field options

PB Class: ○ Node label ● Grammatical form

I/O: 01110111001
Ethernet LAN
Unix Server
TCP/IP
Sub-system 1P

Knowledge: 01010101011
EDI/XML
Mathematics
Gen. faculty
Chemistry

Language: 01011100001
Chinese
English
Jazz
Lewis struct.

Syntactical Level: 01110010100
0 - Byte
1 - Phoneme
2 - Word
3 - Phrase
4 - Clause

Gram. Form: 01111101101
Noun
Verb
Adjective
Conjunction

Gram. Form Var: 01011101001
Trans. verb
Intrans. verb
Copular verb
Coverb

Sub-gram. Form x: 01111101000
1st pers. sing.
1st pers. plural
3rd pers. sing.
3rd pers. plural

Sub-gram. Form y: 01110101111
Present tense
Past tense
Future tense
Past per. tense

Root Word: 01011101011
Run
Hop
Skip
Jump

Root Word Var: 01010111101
Move quickly
Tear
Conduct
Glut

Display Protocol: 01010111111
Graphics sys. 1
Graphics sys. 2
Com interface
Basic object

Root Word ID: 01110001001
Auto
Manual

Fig. 36

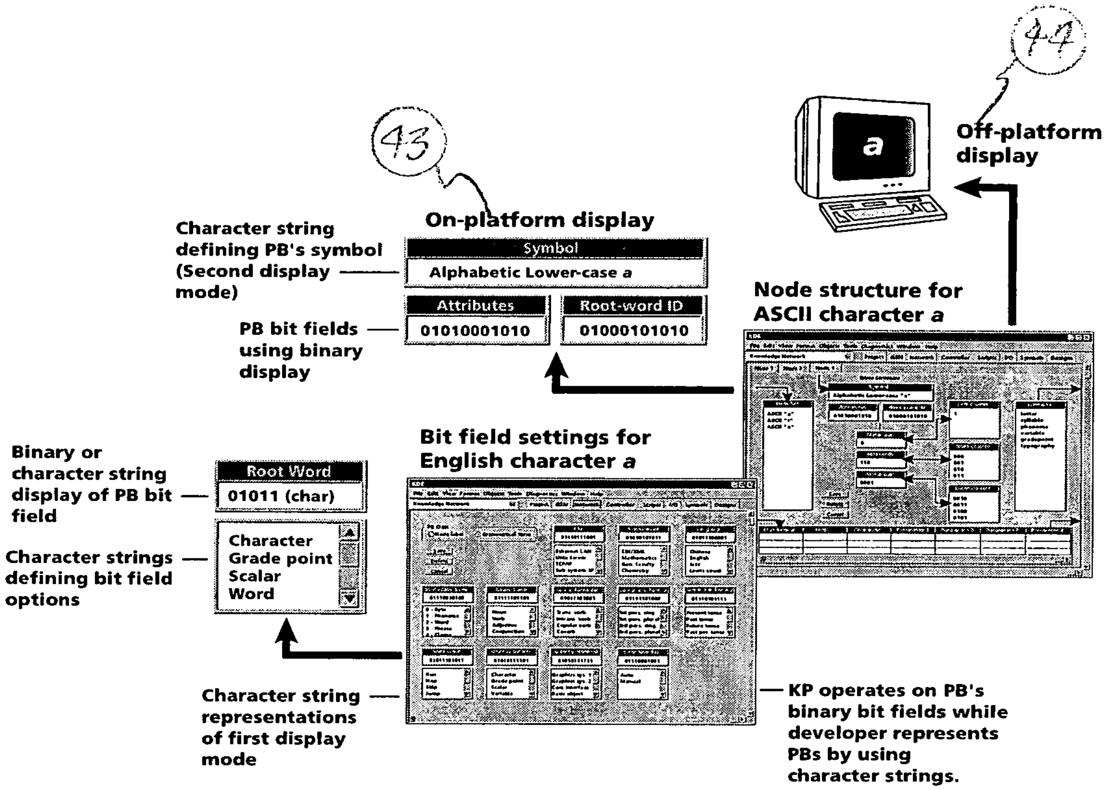
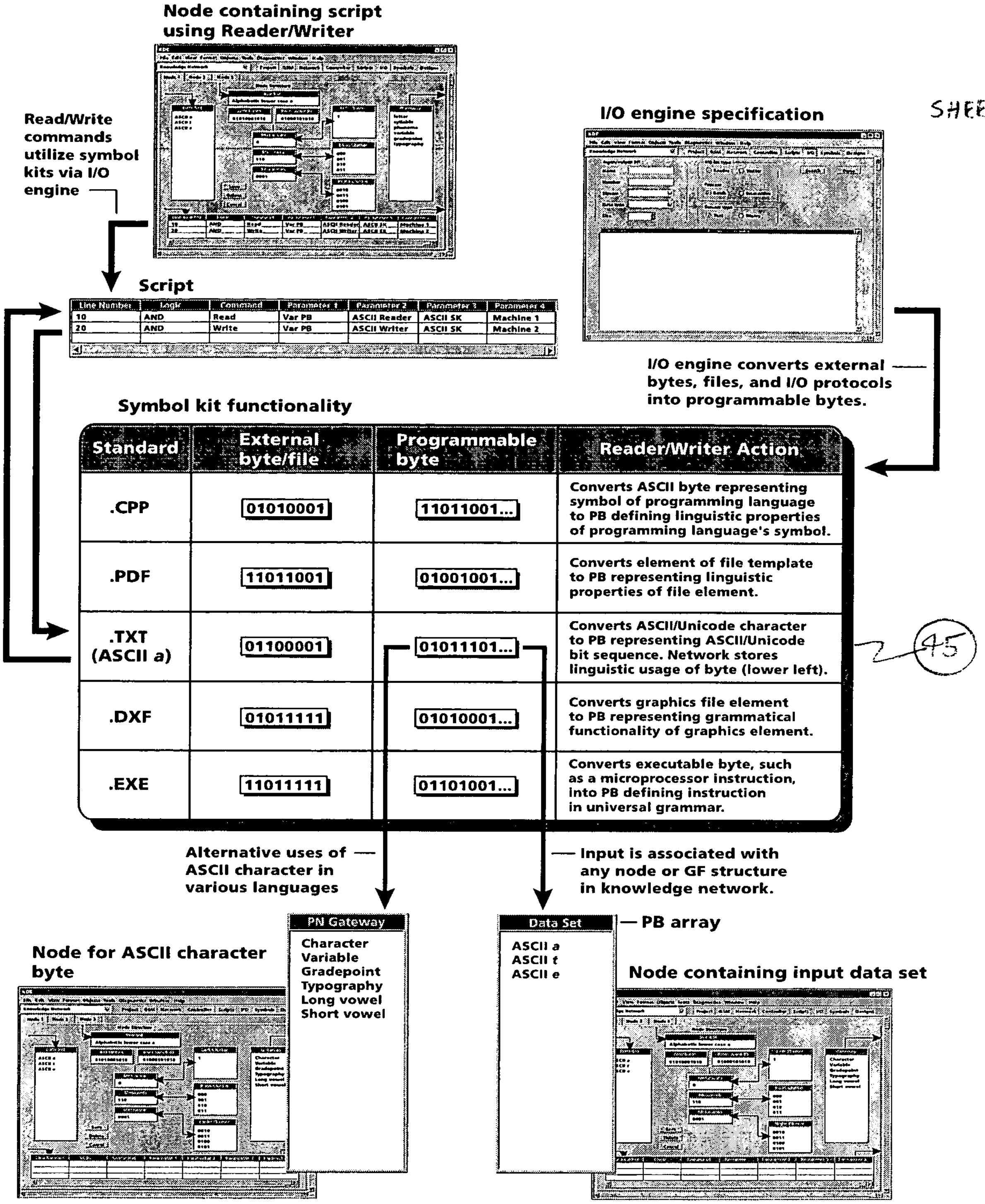

44
Off-platform display
a
43
On-platform display
Character string defining PB's symbol (Second display mode)
Symbol
Alphabetic Lower-case a
Attributes
01010001010
Root-word ID
01000101010
PB bit fields using binary display
Node structure for ASCII character a
Bit field settings for English character a
Binary or character string display of PB bit field
Root Word
01011 (char)
Character strings defining bit field options
Character
Grade point
Scalar
Word
Character string representations of first display mode
KP operates on PB's binary bit fields while developer represents PBs by using character strings.

Fig. 37

SHEET 39 of 140



Node containing script using Reader/Writer

Read/Write commands utilize symbol kits via I/O engine

I/O engine specification

Script

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Read	Var PB	ASCII Reader	ASCII SK	Machine 1
20	AND	Write	Var PB	ASCII Writer	ASCII SK	Machine 2

I/O engine converts external bytes, files, and I/O protocols into programmable bytes.

Symbol kit functionality

Standard	External byte/file	Programmable byte	Reader/Writer Action
.CPP	01010001	11011001...	Converts ASCII byte representing symbol of programming language to PB defining linguistic properties of programming language's symbol.
.PDF	11011001	01001001...	Converts element of file template to PB representing linguistic properties of file element.
.TXT (ASCII *a*)	01100001	01011101...	Converts ASCII/Unicode character to PB representing ASCII/Unicode bit sequence. Network stores linguistic usage of byte (lower left).
.DXF	01011111	01010001...	Converts graphics file element to PB representing grammatical functionality of graphics element.
.EXE	11011111	01101001...	Converts executable byte, such as a microprocessor instruction, into PB defining instruction in universal grammar.

Alternative uses of ASCII character in various languages

Input is associated with any node or GF structure in knowledge network.

Node for ASCII character byte

PN Gateway
- Character
- Variable
- Gradepoint
- Typography
- Long vowel
- Short vowel

Data Set
- ASCII *a*
- ASCII *t*
- ASCII e

PB array

Node containing input data set

Fig. 38

SHEET 140 OF 140

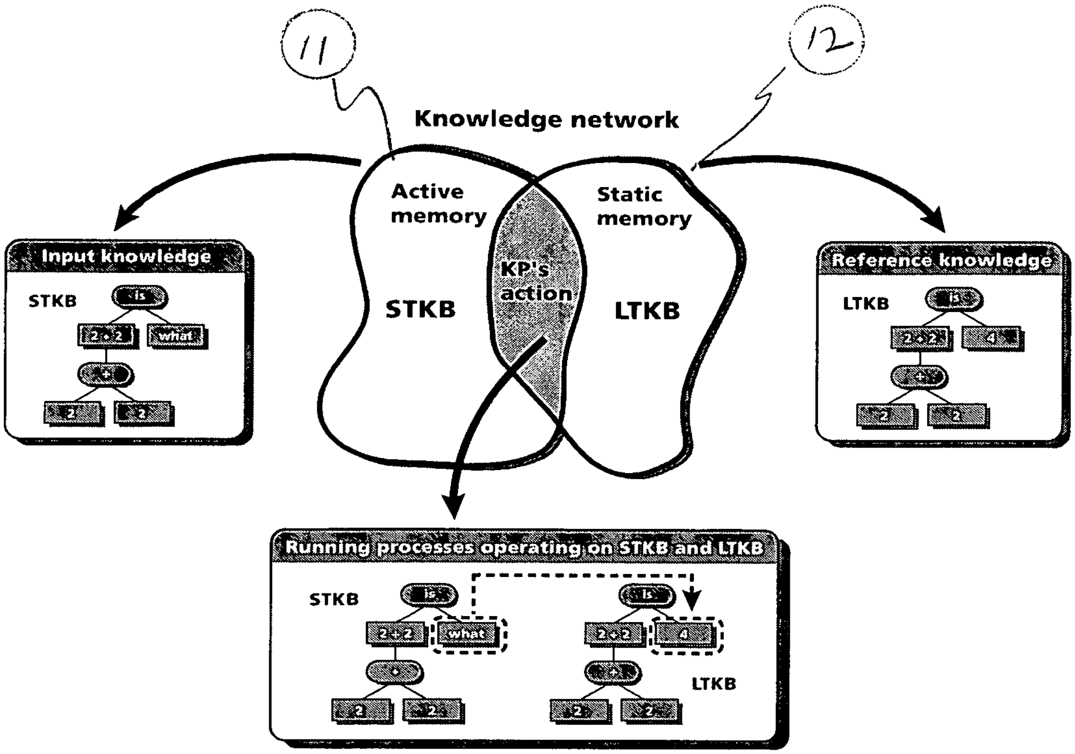


Fig. 39

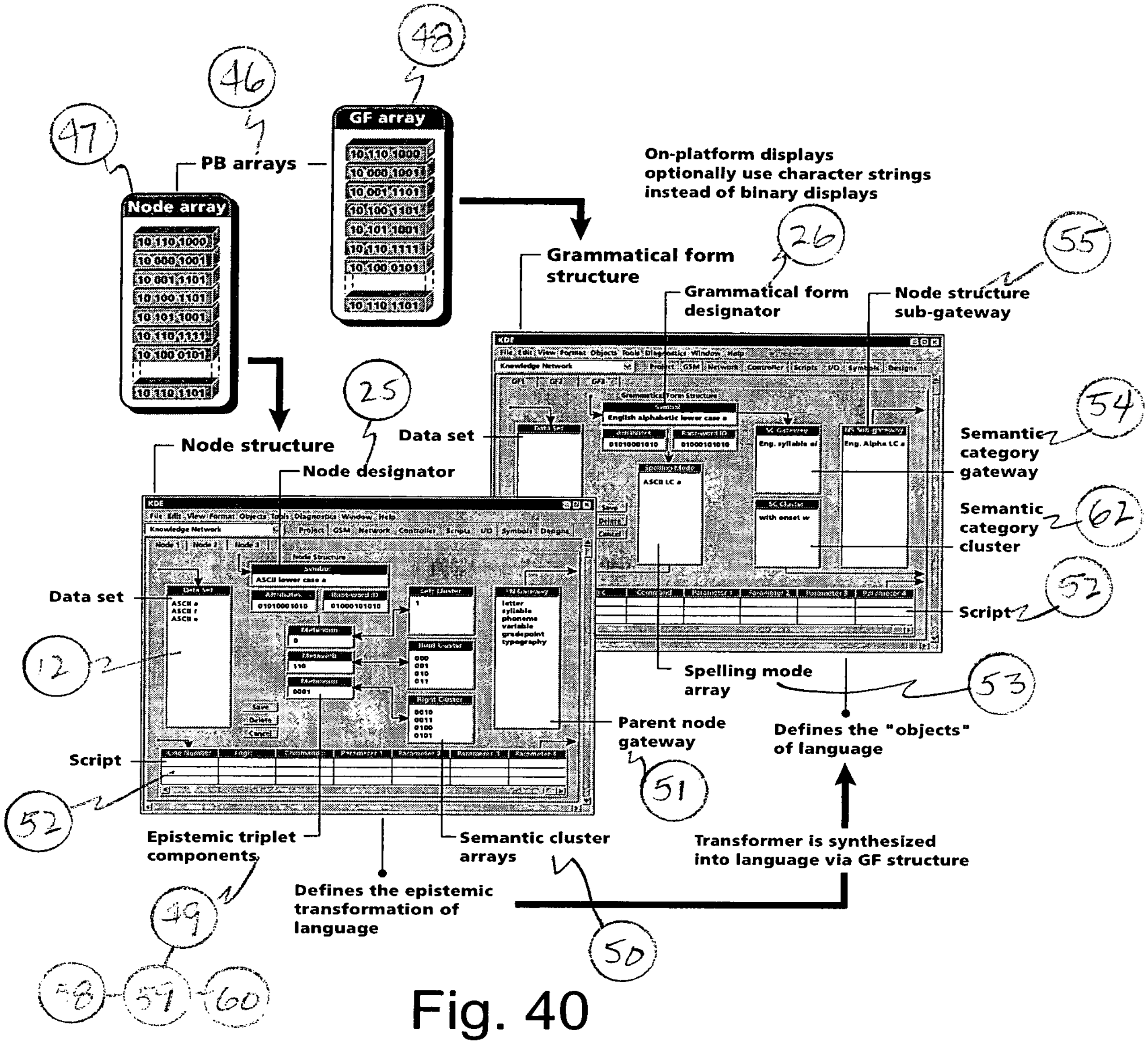

47
46
48
PB arrays
Node array
GF array
10 110 1000
10 000 1001
10 001 1101
10 100 1101
10 101 1001
10 110 1111
10 100 0101
10 110 1101
On-platform displays optionally use character strings instead of binary displays
Grammatical form structure
26
55
Grammatical form designator
Node structure sub-gateway
25
Data set
Semantic category gateway
54
Semantic category cluster
62
Node structure
Node designator
Script
52
Data set
12
Spelling mode array
53
Parent node gateway
51
Defines the "objects" of language
Script
52
Epistemic triplet components
Semantic cluster arrays
Transformer is synthesized into language via GF structure
Defines the epistemic transformation of language
49
50
58
59
60
KDE
Knowledge Network
Node Structure
Symbol
ASCII lower case a
Attributes
01010001010
Root-word ID
01000101010
Data Set
ASCII a
ASCII i
ASCII e
Metanoun
0
Metaverb
110
Metanoun
0001
Left Cluster
1
Head Cluster
000
001
010
011
Right Cluster
0010
0011
0100
0101
PN Gateway
letter
syllable
phoneme
variable
grade point
typography
Save
Delete
Cancel
Line Number
Logic
Command
Parameter 1
Parameter 2
Parameter 3
Parameter 4
Grammatical Form Structure
English alphabetic lower case a
Spelling Mode
ASCII LC a
SC Gateway
Eng. syllable of
SC Cluster
with onset w
NS Sub-gateway
Eng. Alpha LC a

Fig. 40

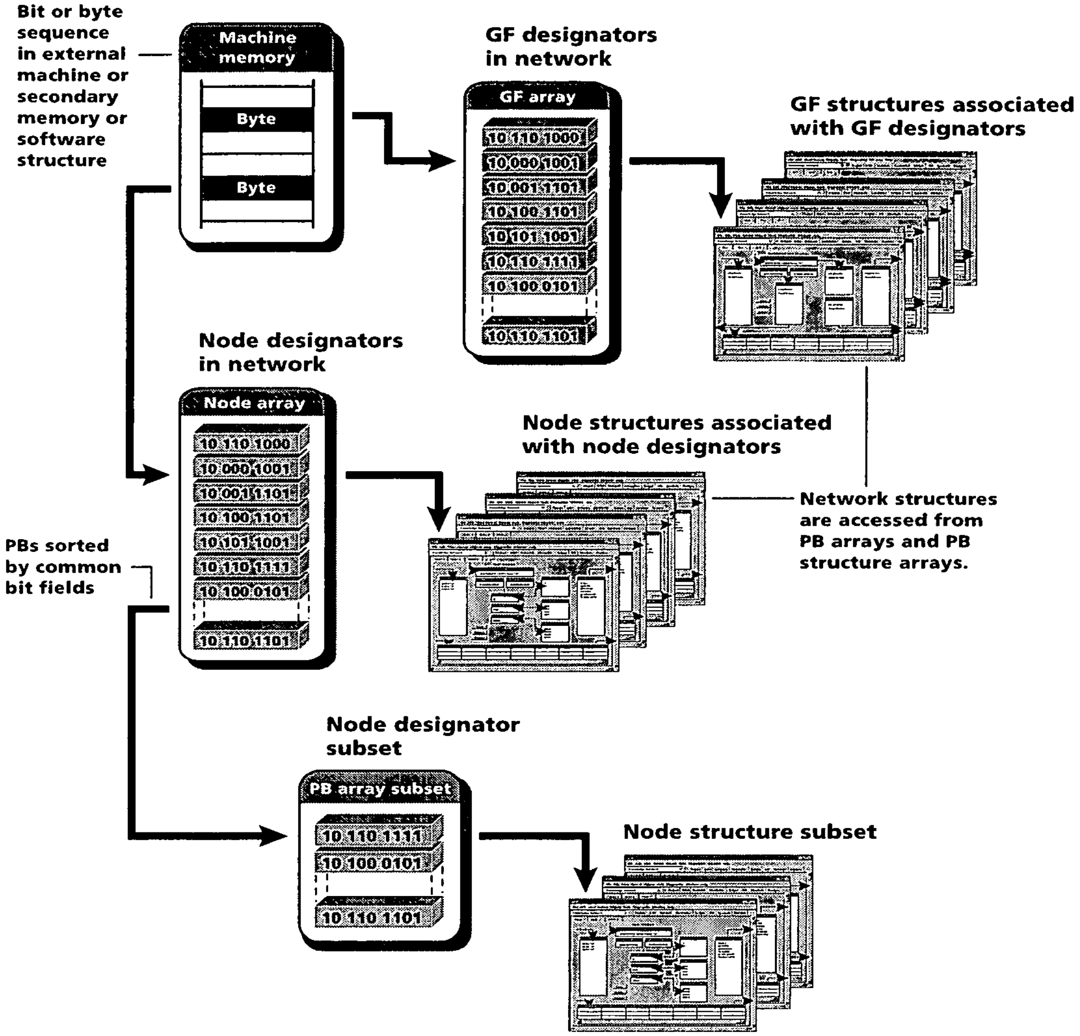
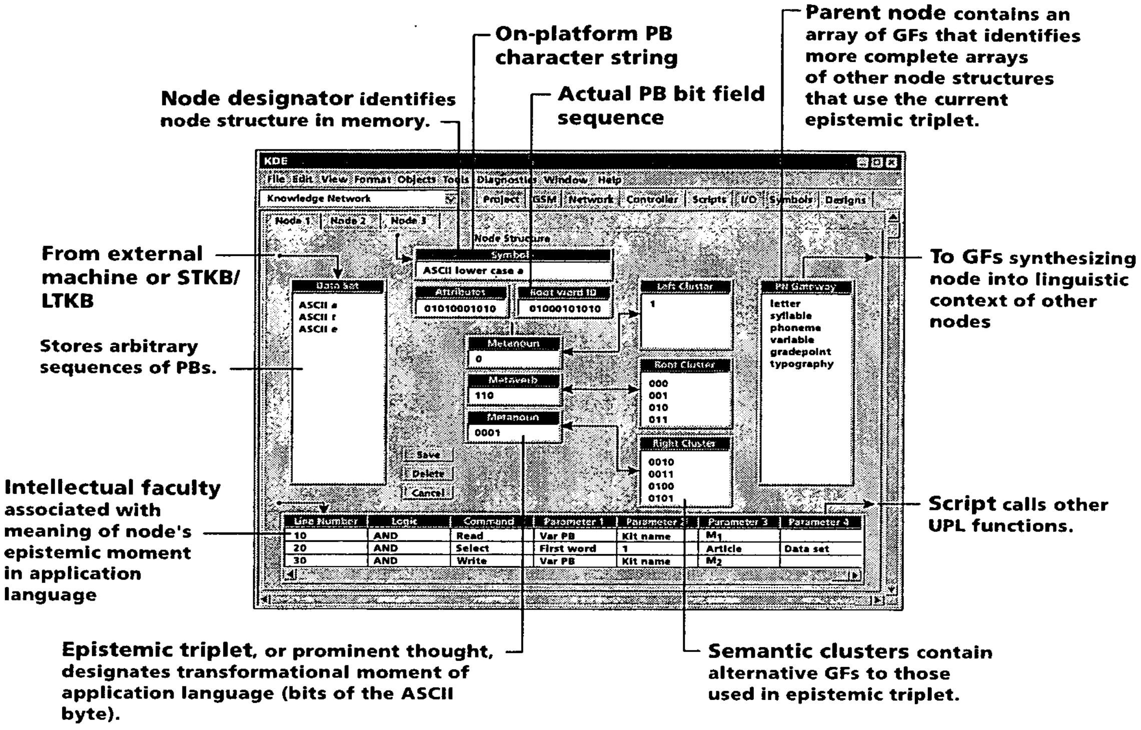

Bit or byte sequence in external machine or secondary memory or software structure
Machine memory
Byte
Byte
GF designators in network
GF array
10 110 1000
10 000 1001
10 001 1101
10 100 1101
10 101 1001
10 110 1111
10 100 0101
10 110 1101
GF structures associated with GF designators
Node designators in network
Node array
10 110 1000
10 000 1001
10 001 1101
10 100 1101
10 101 1001
10 110 1111
10 100 0101
10 110 1101
Node structures associated with node designators
Network structures are accessed from PB arrays and PB structure arrays.
PBs sorted by common bit fields
Node designator subset
PB array subset
10 110 1111
10 100 0101
10 110 1101
Node structure subset

Fig. 41


KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
Node 1 Node 2 Node 3
Node Structure
Save
Delete
Cancel

Symbol: ASCII lower case a

Attributes: 01010001010

Root word ID: 01000101010

Data Set:
ASCII a
ASCII r
ASCII e

Metanoun: 0

Metaverb: 110

Metanoun: 0001

Left Cluster:
1

Root Cluster:
000
001
010
011

Right Cluster:
0010
0011
0100
0101

PB Gateway:
letter
syllable
phoneme
variable
gradepoint
typography

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Read	Var PB	Kit name	M_1	
20	AND	Select	First word	1	Article	Data set
30	AND	Write	Var PB	Kit name	M_2	

Node designator identifies node structure in memory.

On-platform PB character string

Actual PB bit field sequence

Parent node contains an array of GFs that identifies more complete arrays of other node structures that use the current epistemic triplet.

From external machine or STKB/LTKB

Stores arbitrary sequences of PBs.

To GFs synthesizing node into linguistic context of other nodes

Intellectual faculty associated with meaning of node's epistemic moment in application language

Script calls other UPL functions.

Epistemic triplet, or prominent thought, designates transformational moment of application language (bits of the ASCII byte).

Semantic clusters contain alternative GFs to those used in epistemic triplet.

Fig. 42

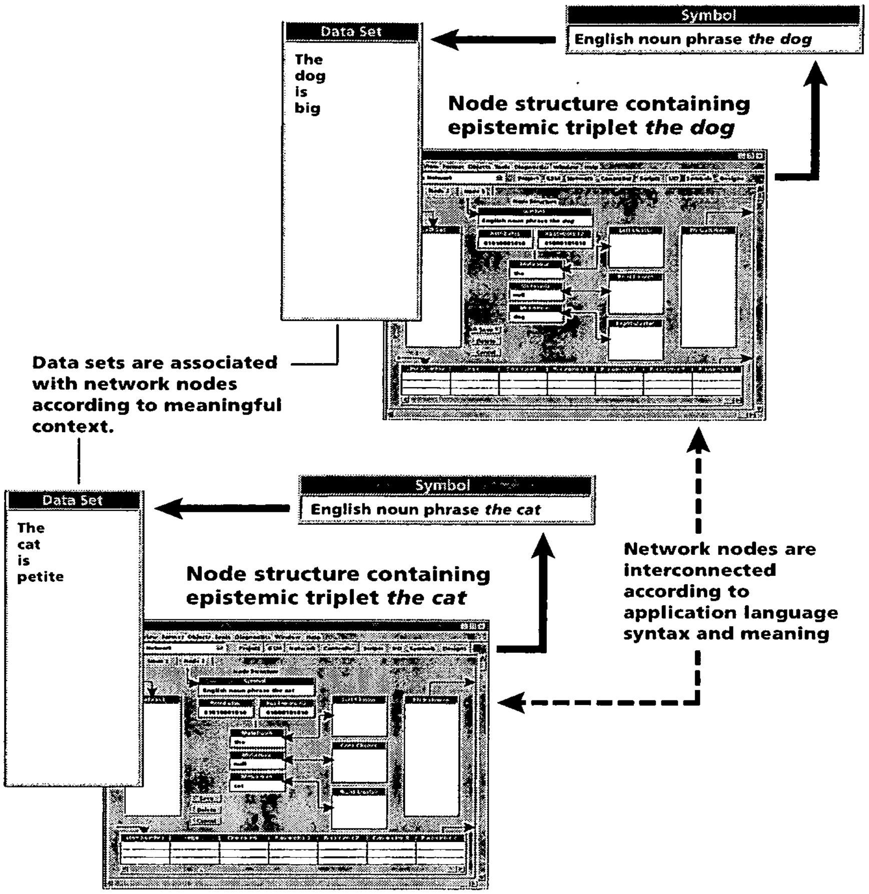

Data Set
The
dog
is
big
Symbol
English noun phrase *the dog*
Node structure containing epistemic triplet *the dog*
Data sets are associated with network nodes according to meaningful context.
Data Set
The
cat
is
petite
Symbol
English noun phrase *the cat*
Node structure containing epistemic triplet *the cat*
Network nodes are interconnected according to application language syntax and meaning

Fig. 43

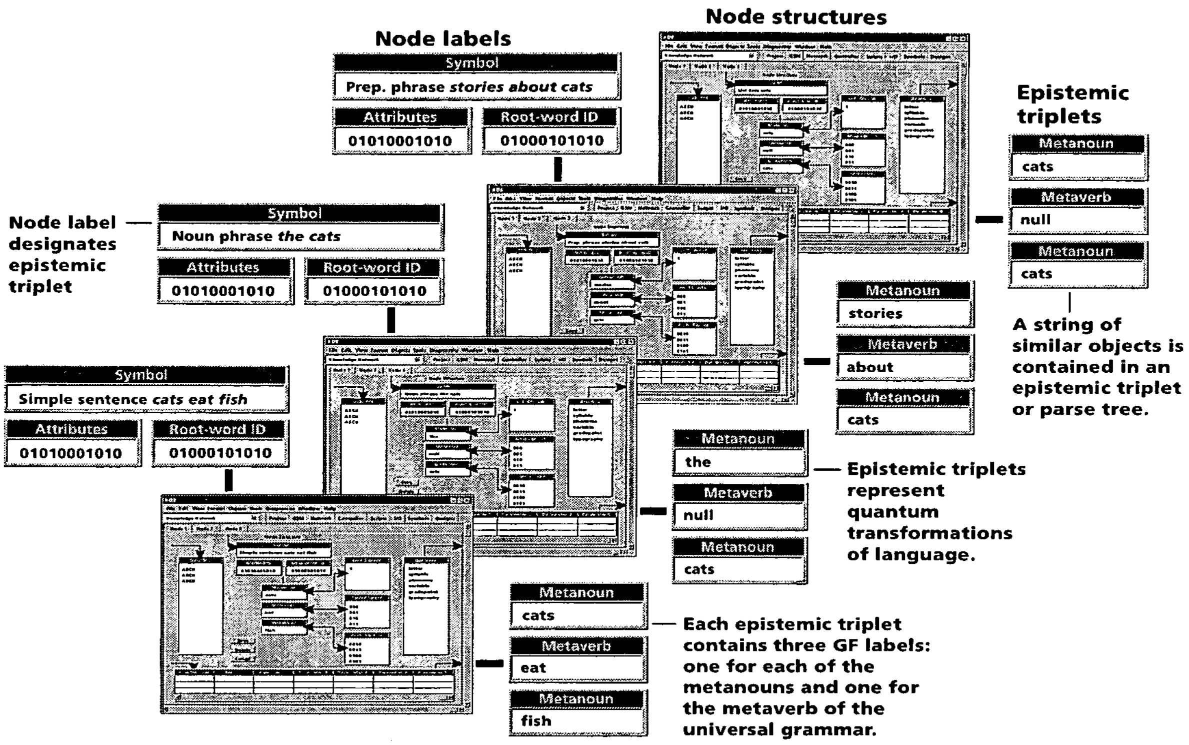

Node labels
Node structures
Symbol
Prep. phrase *stories about cats*
Attributes
01010001010
Root-word ID
01000101010
Epistemic triplets
Metanoun
cats
Metaverb
null
Metanoun
cats
A string of similar objects is contained in an epistemic triplet or parse tree.
Node label designates epistemic triplet
Symbol
Noun phrase *the cats*
Attributes
01010001010
Root-word ID
01000101010
Metanoun
stories
Metaverb
about
Metanoun
cats
Symbol
Simple sentence *cats eat fish*
Attributes
01010001010
Root-word ID
01000101010
Metanoun
the
Metaverb
null
Metanoun
cats
Epistemic triplets represent quantum transformations of language.
Metanoun
cats
Metaverb
eat
Metanoun
fish
Each epistemic triplet contains three GF labels: one for each of the metanouns and one for the metaverb of the universal grammar.

Fig. 44

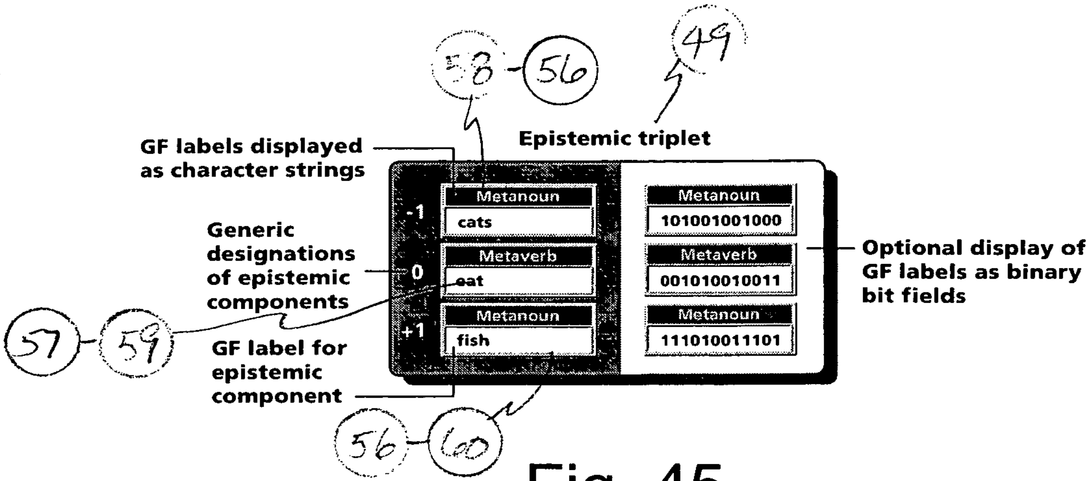
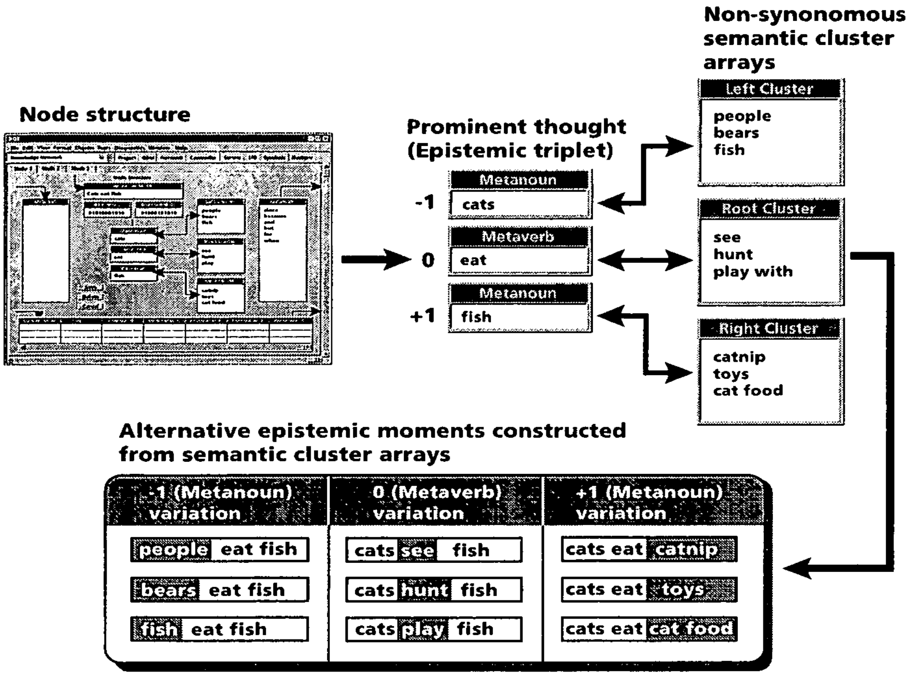

58-56
49
Epistemic triplet
GF labels displayed as character strings
-1
Metanoun
cats
Generic designations of epistemic components
0
Metaverb
eat
+1
Metanoun
fish
57-59
GF label for epistemic component
56-60
Metanoun
101001001000
Metaverb
001010010011
Metanoun
111010011101
Optional display of GF labels as binary bit fields

Fig. 45


Non-synonomous semantic cluster arrays
Node structure
Prominent thought (Epistemic triplet)
Left Cluster
people
bears
fish
-1
Metanoun
cats
Root Cluster
see
hunt
play with
0
Metaverb
eat
+1
Metanoun
fish
Right Cluster
catnip
toys
cat food
Alternative epistemic moments constructed from semantic cluster arrays

-1 (Metanoun) variation	0 (Metaverb) variation	+1 (Metanoun) variation
people eat fish	cats see fish	cats eat catnip
bears eat fish	cats hunt fish	cats eat toys
fish eat fish	cats play fish	cats eat cat food

Fig. 46

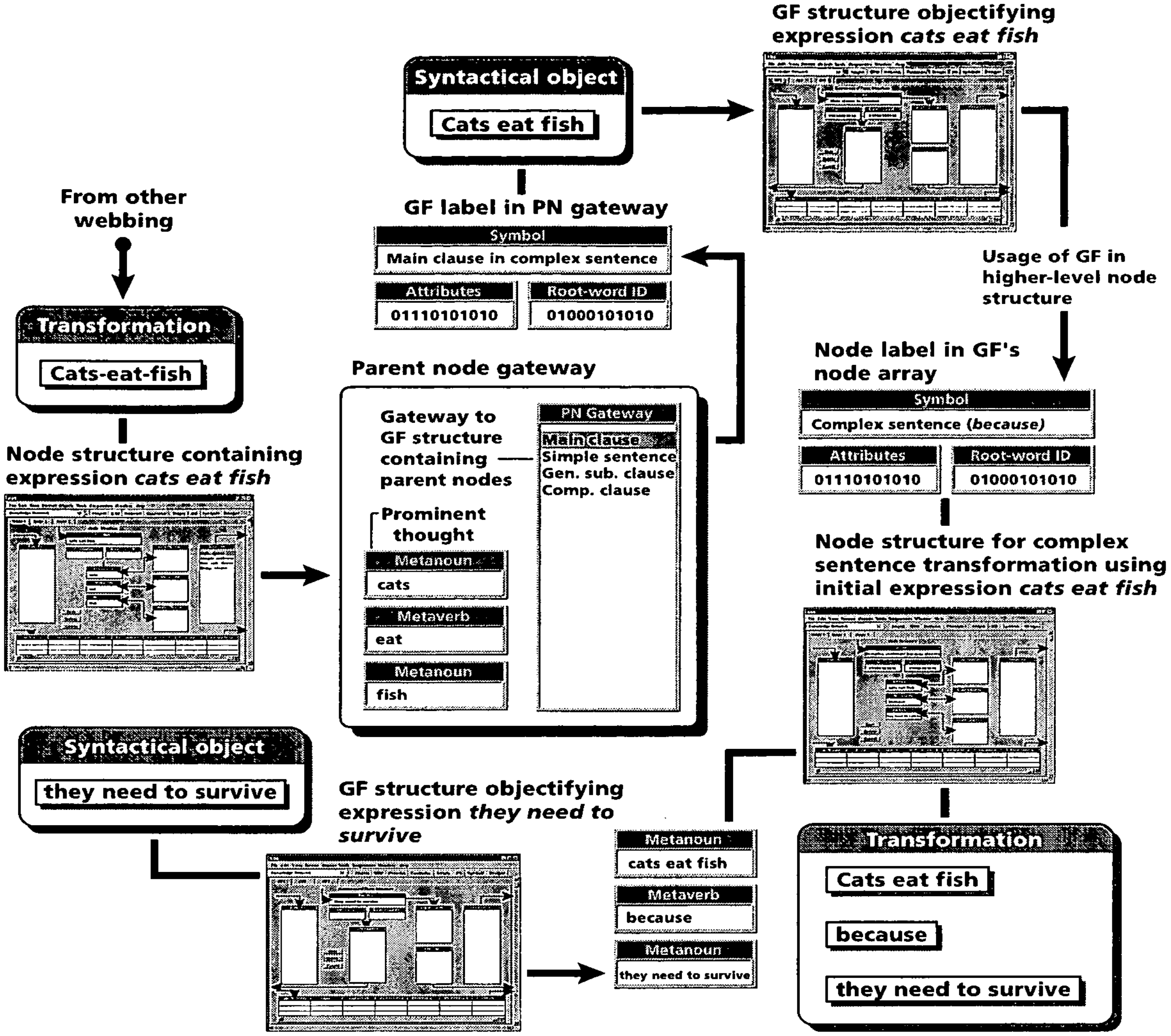
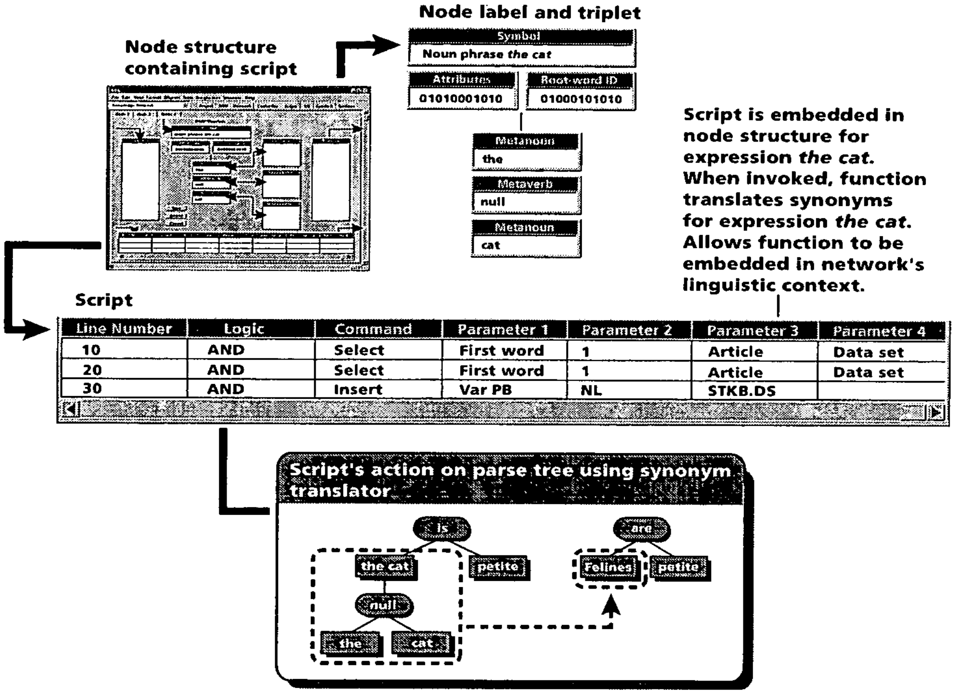

GF structure objectifying
expression *cats eat fish*
Syntactical object
Cats eat fish
GF label in PN gateway
Symbol
Main clause in complex sentence
Attributes
01110101010
Root-word ID
01000101010
From other
webbing
Transformation
Cats-eat-fish
Node structure containing
expression *cats eat fish*
Parent node gateway
Gateway to
GF structure
containing
parent nodes
PN Gateway
Main clause
Simple sentence
Gen. sub. clause
Comp. clause
Prominent
thought
Metanoun
cats
Metaverb
eat
Metanoun
fish
Usage of GF in
higher-level node
structure
Node label in GF's
node array
Symbol
Complex sentence (*because*)
Attributes
01110101010
Root-word ID
01000101010
Node structure for complex
sentence transformation using
initial expression *cats eat fish*
Syntactical object
they need to survive
GF structure objectifying
expression *they need to*
survive
Metanoun
cats eat fish
Metaverb
because
Metanoun
they need to survive
Transformation
Cats eat fish
because
they need to survive

Fig. 47



Script is embedded in node structure for expression ***the cat***. When invoked, function translates synonyms for expression ***the cat***. Allows function to be embedded in network's linguistic context.

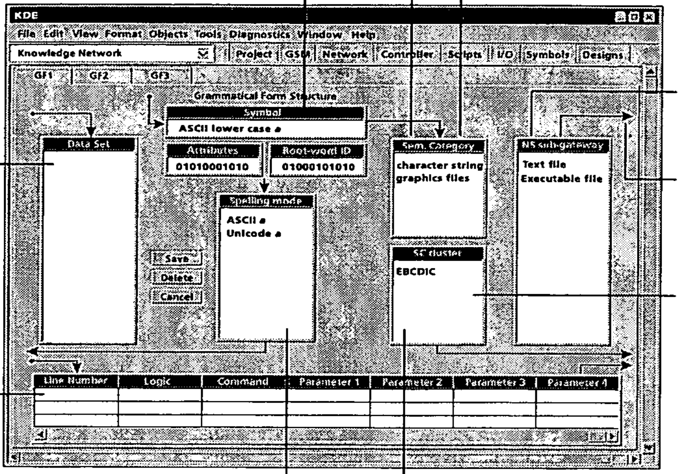
Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	

Fig. 48

Semantic category gateway
(Links to alternative GF structures)

GF labels representing categories
of GF structures semantically
relating to current GF structure

GF label
(Objectifies node structure)



Symbol: ASCII lower case a

Attributes: 01010001010

Root-word ID: 01000101010

Data Set

Spelling mode:
ASCII a
Unicode a

Sem. Category:
character string
graphics files

SC cluster:
EBCDIC

NS sub-gateway:
Text file
Executable file

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4

Node structure
sub-gateway

Data set

To node structures
using current GF
structure in their
epistemic triplets

Semantic category
cluster

Script relating
to current GF
structure

Node structures objectified
by current GF structure.
(Alternative node labels
representing "spelling modes"
of current GF structure)

Particular GF structures relating
to a given GF label in semantic
category gateway (sub-list of GF
labels)

Fig. 49

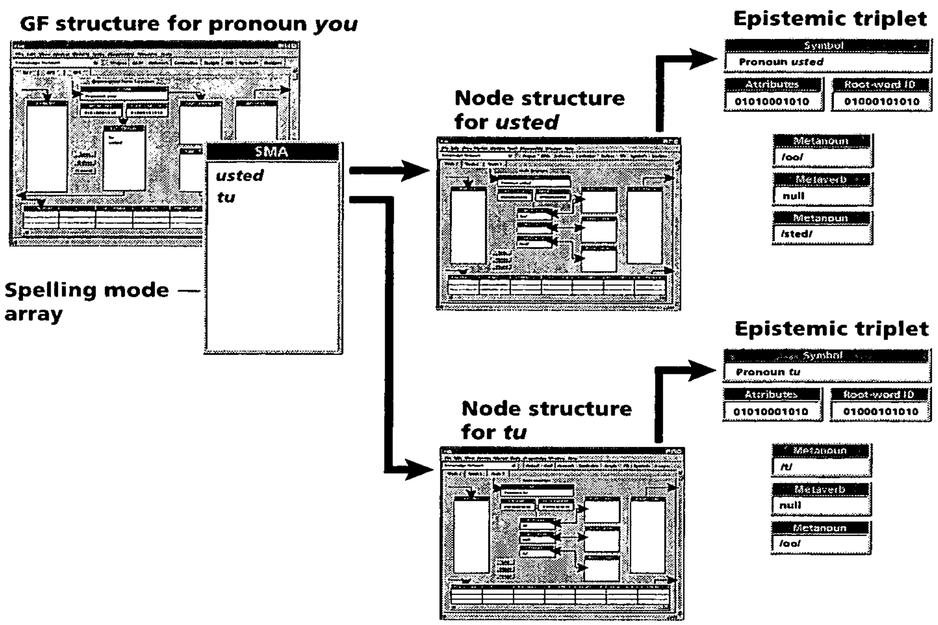
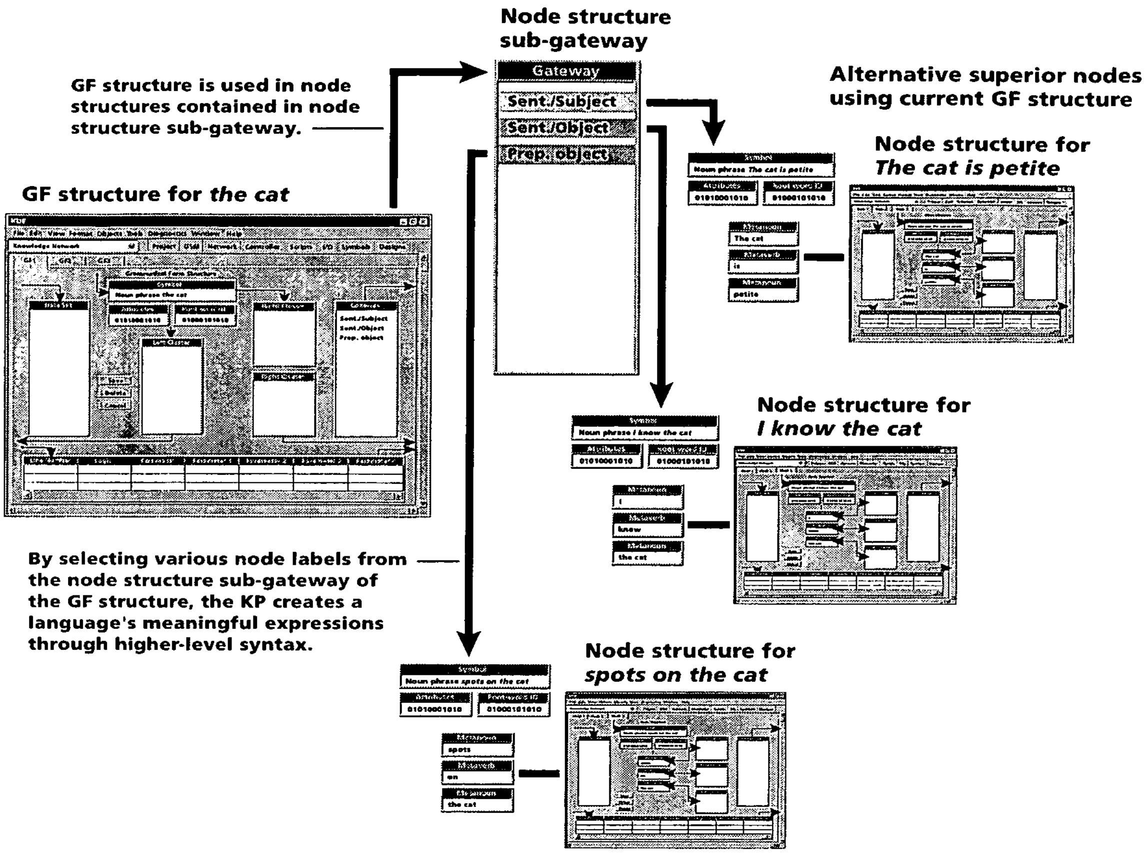

GF structure for pronoun *you*
SMA
usted
tu
Spelling mode array
Node structure for *usted*
Epistemic triplet
Symbol
Pronoun *usted*
Attributes
Root-word ID
01010001010
01000101010
Metanoun
/oo/
Metaverb
null
Metanoun
/sted/
Node structure for *tu*
Epistemic triplet
Symbol
Pronoun *tu*
Attributes
Root-word ID
01010001010
01000101010
Metanoun
/t/
Metaverb
null
Metanoun
/oo/

Fig. 50



Node structure sub-gateway

GF structure is used in node structures contained in node structure sub-gateway.

Alternative superior nodes using current GF structure

Node structure for *The cat is petite*

GF structure for *the cat*

Node structure for *I know the cat*

By selecting various node labels from the node structure sub-gateway of the GF structure, the KP creates a language's meaningful expressions through higher-level syntax.

Node structure for *spots on the cat*

Fig. 51

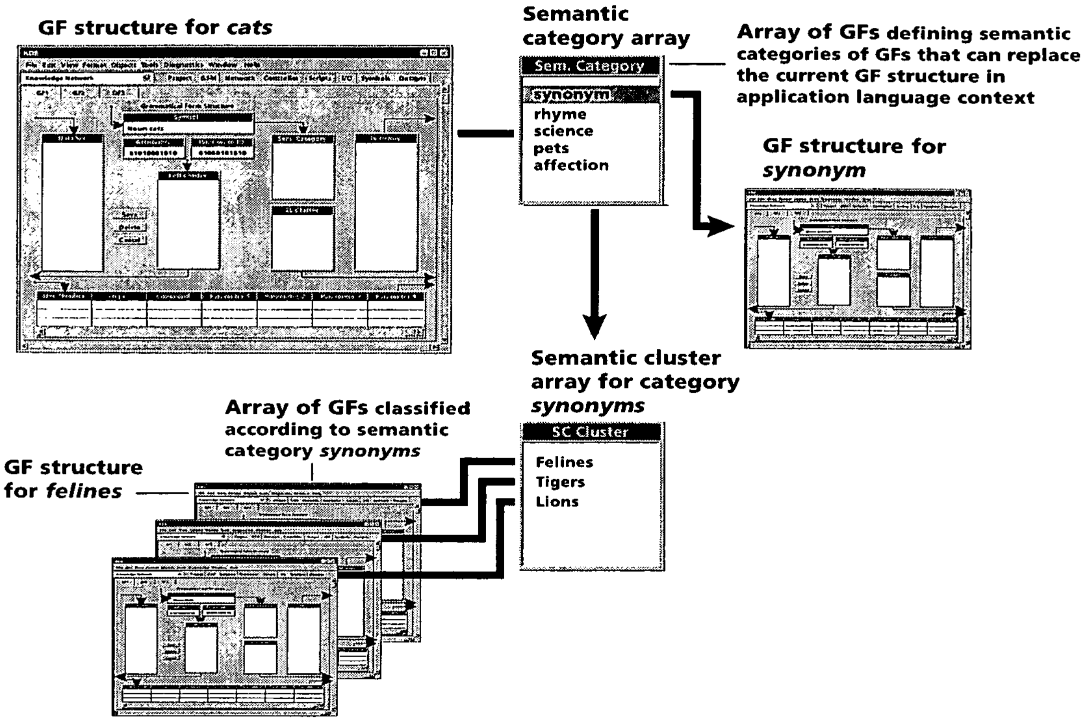

GF structure for *cats*
Semantic category array
Sem. Category
synonym
rhyme
science
pets
affection
Array of GFs defining semantic categories of GFs that can replace the current GF structure in application language context
GF structure for *synonym*
Semantic cluster array for category *synonyms*
SC Cluster
Felines
Tigers
Lions
Array of GFs classified according to semantic category *synonyms*
GF structure for *felines*

Fig. 52

SHEET 54 OF 140

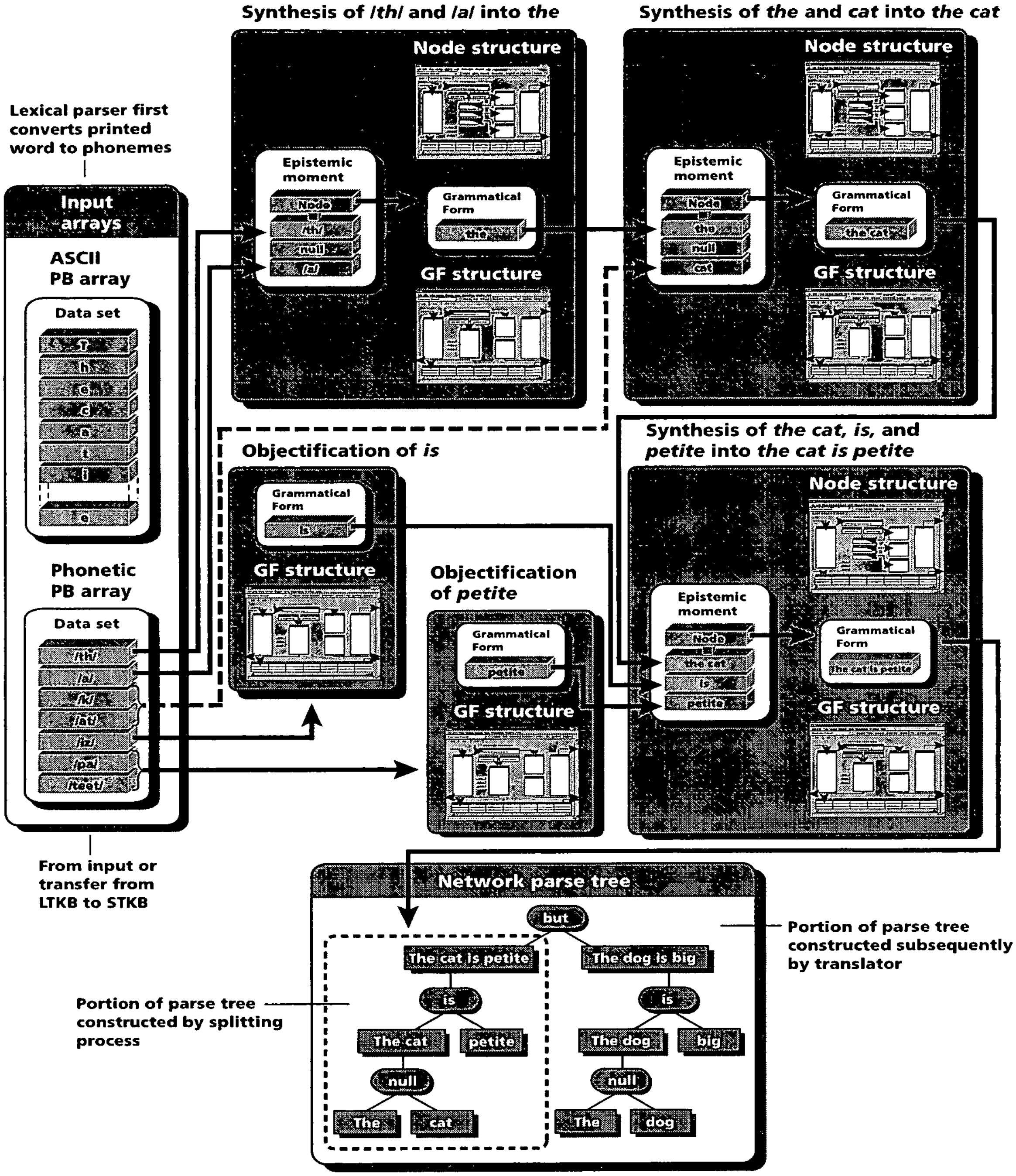


Fig. 53

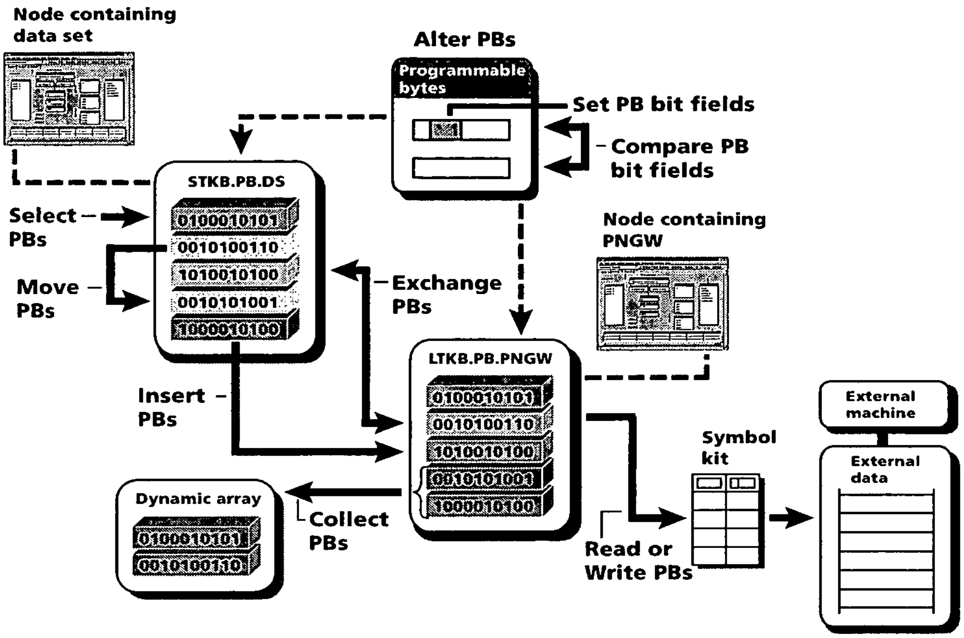

Node containing data set
Alter PBs
Programmable bytes
Set PB bit fields
Compare PB bit fields
STKB.PB.DS
Select PBs
0100010101
0010100110
1010010100
0010101001
1000010100
Move PBs
Exchange PBs
Node containing PNGW
LTKB.PB.PNGW
0100010101
0010100110
1010010100
0010101001
1000010100
Insert PBs
External machine
Symbol kit
External data
Dynamic array
0100010101
0010100110
Collect PBs
Read or Write PBs

Fig. 54

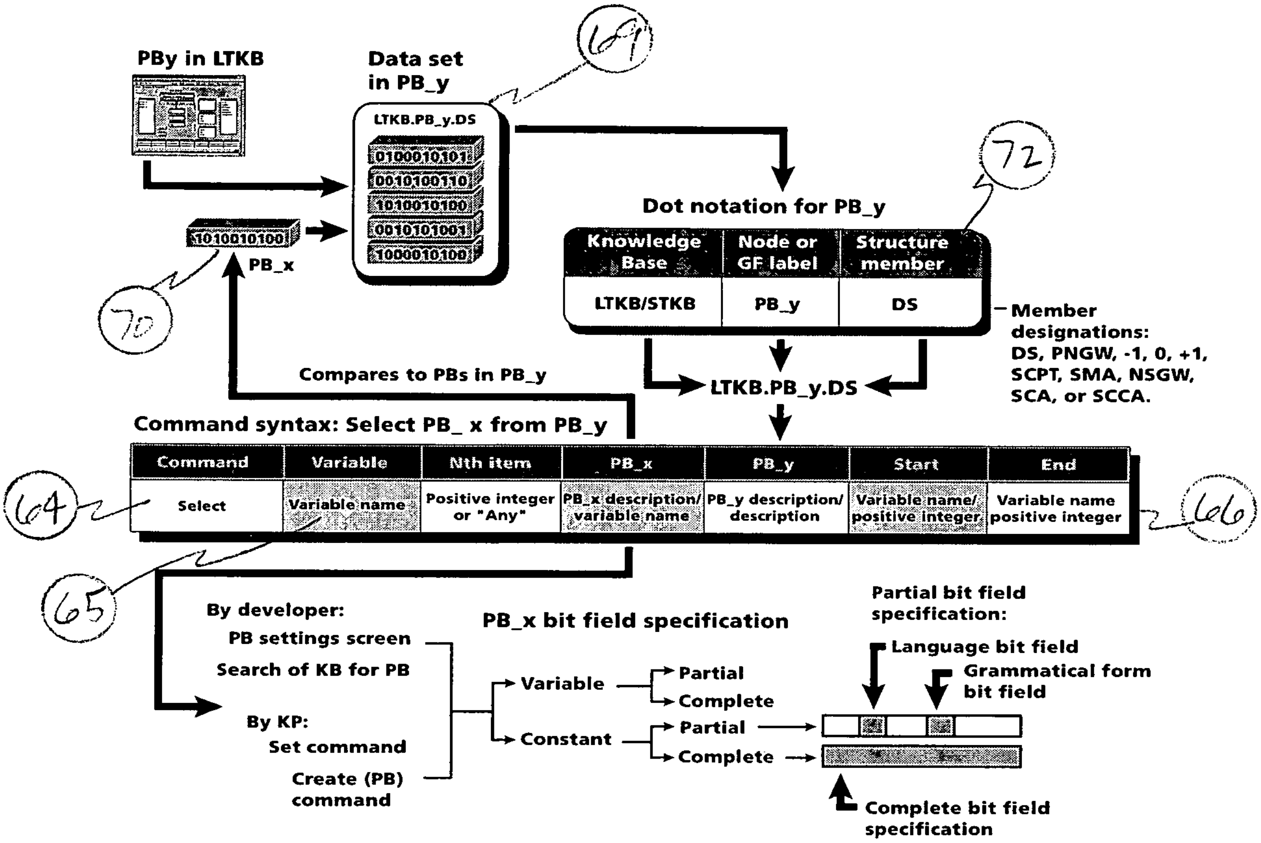

PBy in LTKB
Data set in PB_y
69
LTKB.PB_y.DS
0100010101
0010100110
1010010100
0010101001
1000010100
1010010100
PB_x
70
72
Dot notation for PB_y
Knowledge Base
Node or GF label
Structure member
LTKB/STKB
PB_y
DS
Member designations: DS, PNGW, -1, 0, +1, SCPT, SMA, NSGW, SCA, or SCCA.
LTKB.PB_y.DS
Compares to PBs in PB_y
Command syntax: Select PB_ x from PB_y
Command
Variable
Nth item
PB_x
PB_y
Start
End
Select
Variable name
Positive integer or "Any"
PB_x description/ variable name
PB_y description/ description
Variable name/ positive integer
Variable name positive integer
64
66
65
By developer:
PB settings screen
Search of KB for PB
By KP:
Set command
Create (PB) command
PB_x bit field specification
Variable
Partial
Complete
Constant
Partial
Complete
Partial bit field specification:
Language bit field
Grammatical form bit field
Complete bit field specification

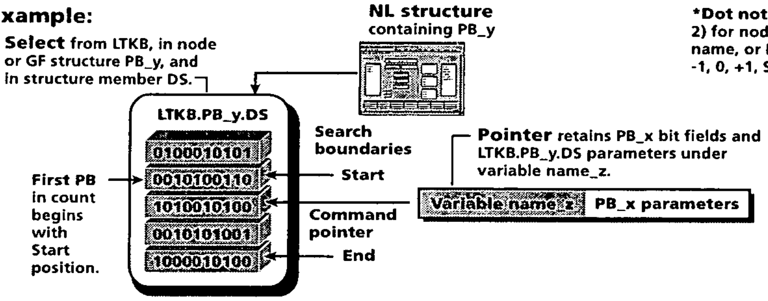
Fig. 55

71

Select PB_ x from PB_ y

Command	Variable	Nth item	PB x	PB y	Start	End
Select	**Variable name_z**	**Positive integer or "Any"**	**PB_x description/ variable name**	**PB_y description/ (Net. structure)**	**Variable name/ positive integer**	**Variable name/ positive integer**
Searches KB structure PB_y for specified PB_x and sets command variable to selected PB_x. **Entry method:** Selection box.	Variable name referenced by other UPL commands in order to access current command's pointer contents. **Entry method:** Alphanumeric character string	Specifies which sequential PB is to be selected when more than one PB in PB_y structure meets PB_x criteria. **Entry method:** Positive integer or the word "Any." ("Any" defaults to 1st item identified.)	Defines PB_x bit fields to be selected using partial or complete bit field specification. **Entry method:** Command variable or PB specified from PB settings screen.	Specifies KB structure PB_y to be searched. **Entry method:** Alphanumeric character string using "dot notation" or command variable.* Selection box for node or GF structure specification.	Specifies starting PB for boundary condition of search. **Entry method:** Variable name or integer	Specifies final PB for boundary condition of search. **Entry method:** Variable name or integer

Example:


Select from LTKB, in node or GF structure PB_y, and in structure member DS.
NL structure containing PB_y
LTKB.PB_y.DS
0100010101
0010100110
1010010100
0010101001
1000010100
First PB in count begins with Start position.
Search boundaries
Start
Command pointer
End
Pointer retains PB_x bit fields and LTKB.PB_y.DS parameters under variable name_z.
Variable name_z
PB_x parameters

*Dot notation: 1) for LTKB or STKB, enter either "LTKB" or "STKB," 2) for node or GF structure, enter PB settings, KB search, variable name, or NL/GF array, and 3) for structure member, enter (DS, PNGW, -1, 0, +1, SCPT, SMA, NSGW, SCA, or SCCA).

Operation: Loads register with PB_x and compares to PBs found in LTKB.PB_ y.DS between Start and End PBs. Comparison proceeds according to any combination of PB bit fields specified in command syntax. Partial comparison executes command on one or more specified bit fields. Sets variable name to selected PB_x in PB_ y.

Fig. 56

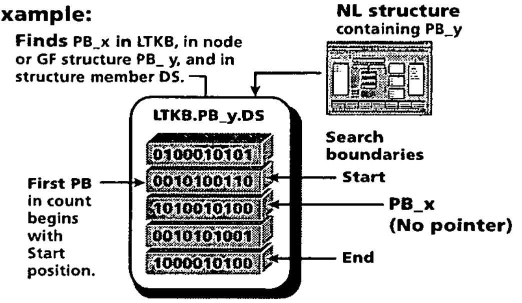
73

SHEET 58 OF 140

Find PB_x from PB_y

Command	Nth item	PB_x	From PB_y	Start	End
Find	**Positive integer/ Any**	**PB_x description/ variable name**	**PB_y description (Net. structure)**	**Variable name/ positive integer**	**Variable name/ positive integer**
Searches KB structure PB_y for specified PB_x and allows UPL function logic to continue if PB_x is present. Discontinues logic if PB_x is not present. **Entry method:** Selection box.	Specifies which sequential PB is to be selected when more than one PB in PB_y structure meets PB_x criteria. **Entry method:** Positive integer or the word "Any." ("Any" defaults to 1st item.)	Defines PB_x bit fields to be searched for using partial or complete bit field specification. **Entry method:** Command variable or PB specified from PB settings screen.	Specifies KB structure PB_y to be searched. **Entry method:** Alphanumeric character string using "dot notation" or command variable. Selection box for node or GF structure specification.*	Specifies starting PB for boundary condition of search. **Entry method:** Variable name or integer.	Specifies final PB for boundary condition of search. **Entry method:** Variable name or integer.

Example:

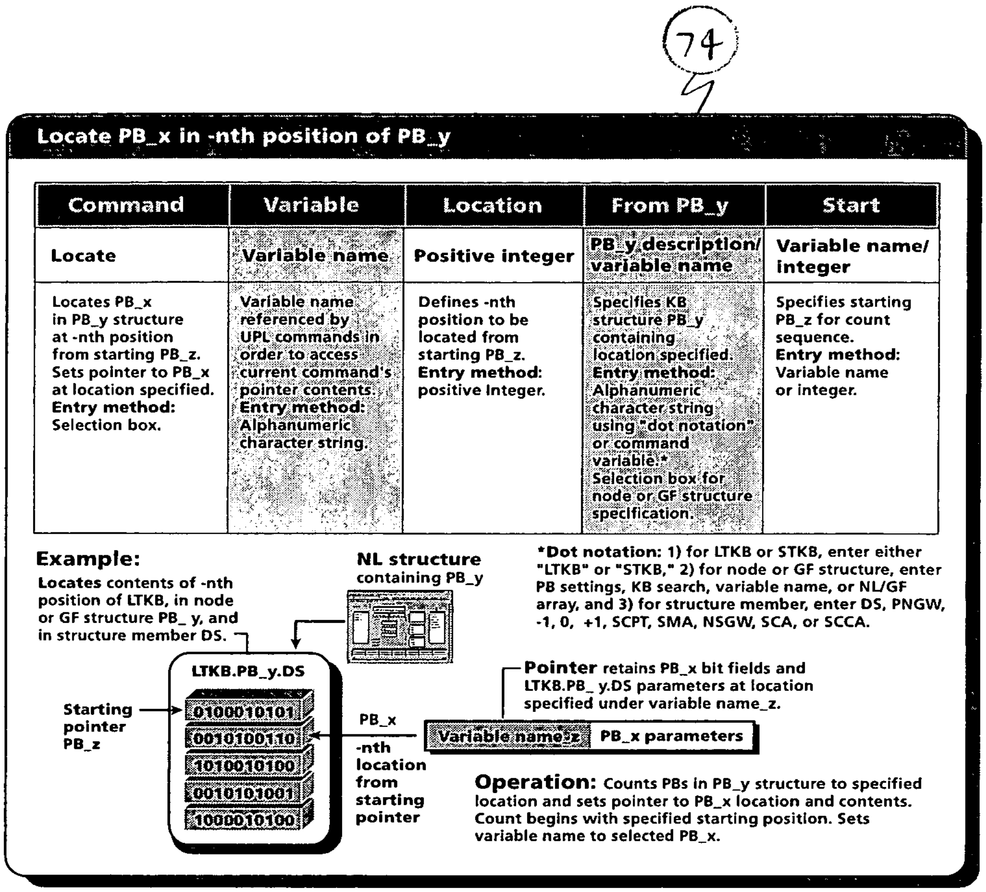


Finds PB_x in LTKB, in node or GF structure PB_ y, and in structure member DS.

***Dot notation:** 1) for LTKB or STKB, enter either "LTKB" or "STKB," 2) for node or GF structure, enter PB settings, KB search, variable name, or NL/GF array, and 3) for structure member, enter DS, PNGW, -1, 0, +1, SCPT, SMA, NSGW, SCA, or SCCA.

Operation: Loads register with PB_x and compares to PBs found in LTKB.PB_ y.DS between Start and End PBs. Comparison proceeds according to any combination of PB bit fields specified in command syntax. Partial comparison executes command on one or more specified bit fields. Determines whether UPL function logic proceeds.

Fig. 57



Locate PB_x in -nth position of PB_y

Command	Variable	Location	From PB_y	Start
Locate	Variable name	Positive integer	PB_y description/ variable name	Variable name/ integer
Locates PB_x in PB_y structure at -nth position from starting PB_z. Sets pointer to PB_x at location specified. **Entry method:** Selection box.	Variable name referenced by UPL commands in order to access current command's pointer contents. **Entry method:** Alphanumeric character string.	Defines -nth position to be located from starting PB_z. **Entry method:** positive Integer.	Specifies KB structure PB_y containing location specified. **Entry method:** Alphanumeric character string using "dot notation" or command variable.* Selection box for node or GF structure specification.	Specifies starting PB_z for count sequence. **Entry method:** Variable name or integer.

Example:
Locates contents of -nth position of LTKB, in node or GF structure PB_ y, and in structure member DS.

NL structure containing PB_y

***Dot notation:** 1) for LTKB or STKB, enter either "LTKB" or "STKB," 2) for node or GF structure, enter PB settings, KB search, variable name, or NL/GF array, and 3) for structure member, enter DS, PNGW, -1, 0, +1, SCPT, SMA, NSGW, SCA, or SCCA.

Pointer retains PB_x bit fields and LTKB.PB_ y.DS parameters at location specified under variable name_z.

Operation: Counts PBs in PB_y structure to specified location and sets pointer to PB_x location and contents. Count begins with specified starting position. Sets variable name to selected PB_x.

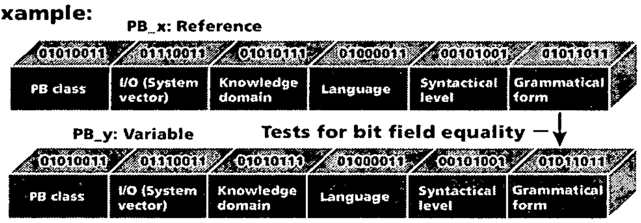
Fig. 58

75

Compare (Test for) PB_x bit fields to PB_y

Command	PB_y	For (PB_x)
Compare (Test for)	**Variable name**	**PB_x description**
Compares variable PB_y to reference PB_x to test for matching bit fields **Entry method:** Selection box.	Specifies PB_y, whose bit fields are to be tested. **Entry method:** Alphanumeric character string.	Defines reference PB_x bit fields to be used for comparison to variable PB_y's bit fields. **Entry method:** PB settings screen.

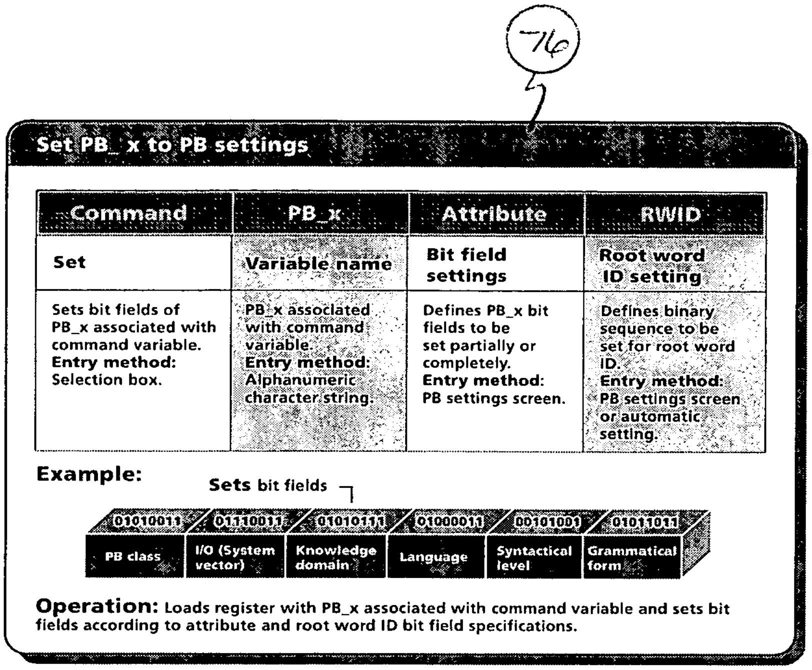
Example:



Operation: Loads registers with PB_x (the reference PB) and PB_y (PB associated with command variable). If PBs match, true condition is returned to command line. If PBs are dissimilar, false condition is returned to command line.

Fig. 59

SHEET 61 OF 140



Set PB_ x to PB settings

Command	PB_x	Attribute	RWID
Set	**Variable name**	**Bit field settings**	**Root word ID setting**
Sets bit fields of PB_x associated with command variable. **Entry method:** Selection box.	PB_x associated with command variable. **Entry method:** Alphanumeric character string.	Defines PB_x bit fields to be set partially or completely. **Entry method:** PB settings screen.	Defines binary sequence to be set for root word ID. **Entry method:** PB settings screen or automatic setting.

Example:

Sets bit fields

PB class	I/O (System vector)	Knowledge domain	Language	Syntactical level	Grammatical form
01010011	01110011	01010111	01000011	00101001	01011011

Operation: Loads register with PB_x associated with command variable and sets bit fields according to attribute and root word ID bit field specifications.

Fig. 60

SHEET 62 OF 140

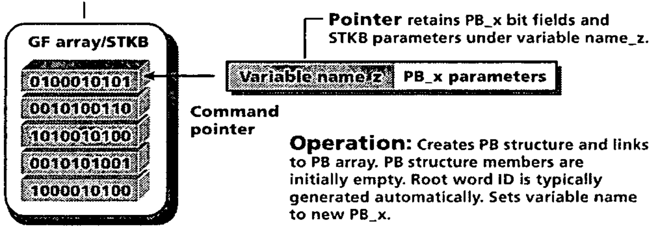
77

Create PB_x in LTKB/STKB

Command	Variable	PB_x	KB_xy
Create	**Variable name_z**	**NL/GF**	**LTKB/STKB**
Creates node or GF structure in LTKB or STKB and assigns new PB_x structure to command variable. **Entry method:** Selection box.	Variable name referenced by other UPL commands in order to access current command's pointer contents. **Entry method:** Alphanumeric character string.	Defines bit fields for PB_x to be created. **Entry method:** PB settings screen.	Defines placement of PB_x in KB once PB_x is created. **Entry method:** Selection box.

Example:

Creates GF structure PB_ x, and places into GF array of STKB



Pointer retains PB_x bit fields and STKB parameters under variable name_z.

Operation: Creates PB structure and links to PB array. PB structure members are initially empty. Root word ID is typically generated automatically. Sets variable name to new PB_x.

Fig. 61

78

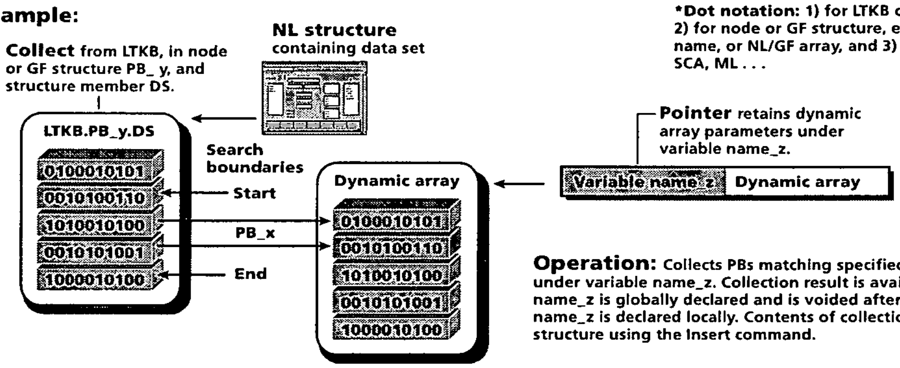
SHEET 63 OF 140

Collect n-many PB_xs from PB_y

Command	Variable	N-many	PB_x	From PB_y	Start	End
Collect	**Variable name_z**	**Positive integer or "All"**	**PB_x description/ variable name**	**PB_y description/ (Net. structure)**	**Variable name/ positive integer**	**Variable name/ positive integer**
Searches KB structure member PB_y and collects PBs meeting PB_x criteria into dynamic command array. Sets command variable (pointer) to array. **Entry method:** Selection box.	Variable name referenced by other UPL commands in order to access current command's pointer contents. **Entry method:** Alphanumeric character string	Specifies the number of PBs to be collected meeting PB_x selection criteria. **Entry method:** Positive integer or the word "All." ("All" collects every item meeting PB_x criteria.)	Defines PB bit fields to be used for collection. Partial or complete bit field specification. **Entry method:** Variable name or PB settings screen.	Specifies KB structure PB_y to be collected from. **Entry method:** Alphanumeric character string using "dot notation" or command variable.* Selection box for node or GF structure specification.	Specifies starting PB for boundary condition for collection. **Entry method:** Variable name or integer.	Specifies final PB for boundary condition for collection. **Entry method:** Variable name or integer

Example:

*Dot notation: 1) for LTKB or STKB, enter either "LTKB" or "STKB" 2) for node or GF structure, enter PB settings, KB search, variable name, or NL/GF array, and 3) for structure member, enter DS, PNGW, SCA, ML . . .



Operation: Collects PBs matching specified PB_x bit fields into dynamic array under variable name_z. Collection result is available to project scripts when variable name_z is globally declared and is voided after UPL function executes when variable name_z is declared locally. Contents of collection are typically inserted into KB structure using the Insert command.

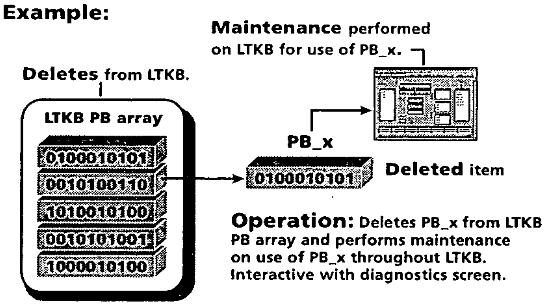
Fig. 62

79

Delete PB_x from KB_xy

Command	PB_x	Maintenance
Delete	PB_x description/ variable name	Check/ No check
Deletes PB_x from LTKB. **Entry method:** Selection box.	Defines PB bit fields to be used for deletion. Partial or complete bit field specification. **Entry method:** Variable name or PB settings screen.	Defines whether auto-maintenance is to be conducted after deletion of PB_x. **Entry method:** Selection box.

Example:



Operation: Deletes PB_x from LTKB PB array and performs maintenance on use of PB_x throughout LTKB. Interactive with diagnostics screen.

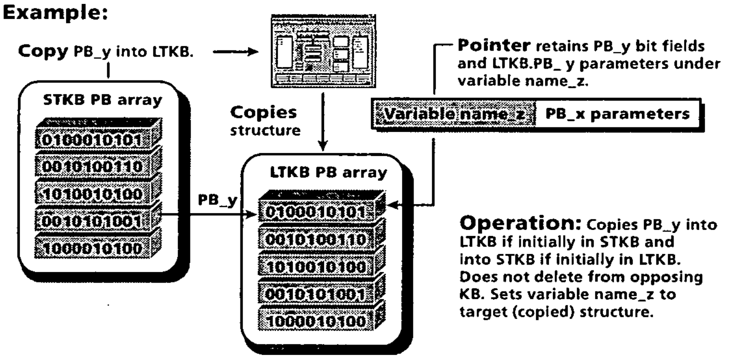
Fig. 63

80

Copy PB_ y into KB_xy

Command	Variable	From PB_y	KB_xy
Copy	**Variable name_z**	**PB description/ variable name**	**LTKB/STKB**
Copies PB_y structure into opposing KB (from LTKB into STKB or from STKB into LTKB). **Entry method:** Selection box.	Variable name referenced by other UPL commands in order to access current command's pointer contents. **Entry method:** Alphanumeric character string.	Defines PB bit fields to be used for copying structure. Partial or complete bit field specification. **Entry method:** Variable name or PB settings screen. Dot notation for structure member.	Specifies KB structure to be copied into. **Entry method:** Selection box.

Example:



Pointer retains PB_y bit fields and LTKB.PB_ y parameters under variable name_z.

Operation: Copies PB_y into LTKB if initially in STKB and into STKB if initially in LTKB. Does not delete from opposing KB. Sets variable name_z to target (copied) structure.

Fig. 64

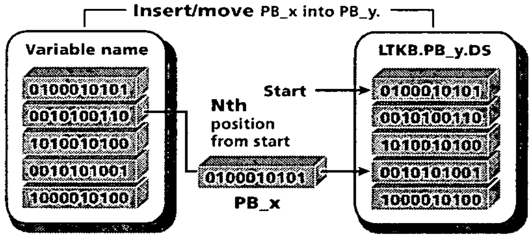
81

SHEET 66 OF 140

Insert (move) PB_ x into PB_ y

Command	PB_x	Into PB_y	Nth	Start
Insert (Move)	**PB description/ variable name**	**PB description/ (Net. structure)**	**Positive integer**	**Variable name/ positive integer**
Inserts or moves PB_x into KB structure member PB_y. Insert command copies PB_x, while Move command copies and removes PB_x. **Entry method:** Selection box.	Defines PB bit fields to be used for insertion/ move. Partial or complete bit field specification. **Entry method:** Variable name or PB settings screen.	Specifies KB structure PB_y to be inserted/moved into. **Entry method:** Alphanumeric character string using "dot notation" or command variable. Selection box for node or GF structure specification.	Specifies offset from starting PB boundary condition for insertion/move. **Entry method:** Alphanumeric Positive integer.	Specifies starting PB for boundary condition of insertion/move. **Entry method:** Variable name or integer for "Nth" item.

Example:



Operation: Insert command copies PB_x from KB structure specified by variable name and inserts into designated location in PB_y structure. Move command performs Insert command action but also removes PB_x from initial structure specified by variable name.

Fig. 65

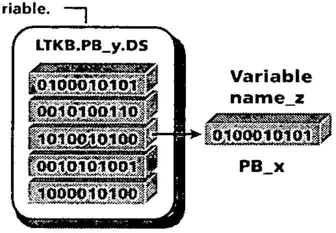
82

Remove PB_x from KB structure member

Command	PB_x
Remove	**Variable name_z**
Removes PB_x from KB structure identified by command variable. **Entry method:** Selection box.	Specifies PB_x to be removed from its KB structure PB_y. **Entry method:** Alphanumeric character string.

Example:

Remove PB_x from structure designated by command variable.



Operation: Removes PB_x from PB_y designated by variable name_z.

Fig. 66

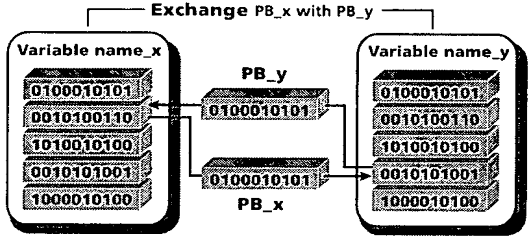
SHEET 68 OF 140

83

Exchange PB_x with PB_y

Command	PB	PB
Exchange	**Variable name_x**	**Variable name_y**
Searches KB structure member for specified PB_x and sets command pointer to selected PB, using command variable name, for subsequent reference by other commands. **Entry method:** Selection box.	Specifies PB_x to be removed from KB structure specified by variable name_x and inserted into KB structure specified by variable name_y at location PB_y. **Entry method:** Alphanumeric character string.	Specifies PB_y to be removed from KB structure specified by variable name_y and inserted into KB structure specified by variable name_x at location PB_x. **Entry method:** Alphanumeric character string.

Example:



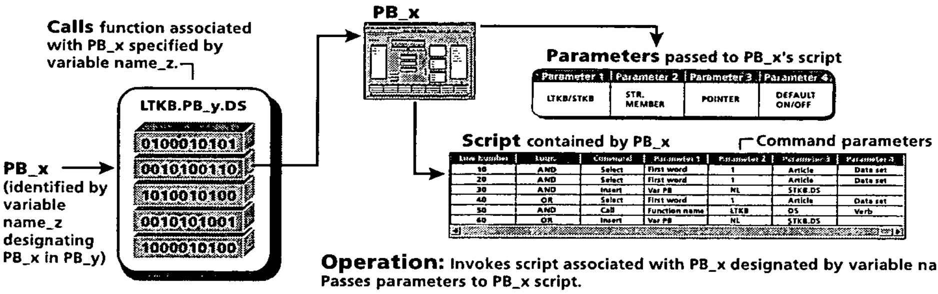
Operation: Swaps PB_x with PB_y by performing consecutive moves on PB_x and PB_y.

Fig. 67

Call function PB_x

Command	Function PB_x	Parameter 1	Parameter 2	Parameter 3	Default
Call	Variable name_z	Variable name	Variable name	Variable name	On/Off
Invokes UPL function associated with KB structure member PB_x; then passes specified parameters, and receives result of invoked function. **Entry method:** Selection box.	Specifies KB structure member PB_x containing invoked function. **Entry method:** Alphanumeric character string.	Specifies KB structure Parameter 1 (usually LTKB/STKB). **Entry method:** Alphanumeric character string.	Specifies KB structure Parameter 1 (usually a pointer in Parameter 1). **Entry method:** Alphanumeric character string.	Specifies KB structure Parameter 1 (usually a pointer in Parameter 1). **Entry method:** Alphanumeric character string.	Specifies preferred or default action for command logic. **Entry method:** Selection box.

Example:



Calls function associated with PB_x specified by variable name_z.
LTKB.PB_y.DS
0100010101
0010100110
1010010100
0010101001
1000010100
PB_x
(identified by variable name_z designating PB_x in PB_y)
PB_x
Parameters passed to PB_x's script
Script contained by PB_x
Command parameters

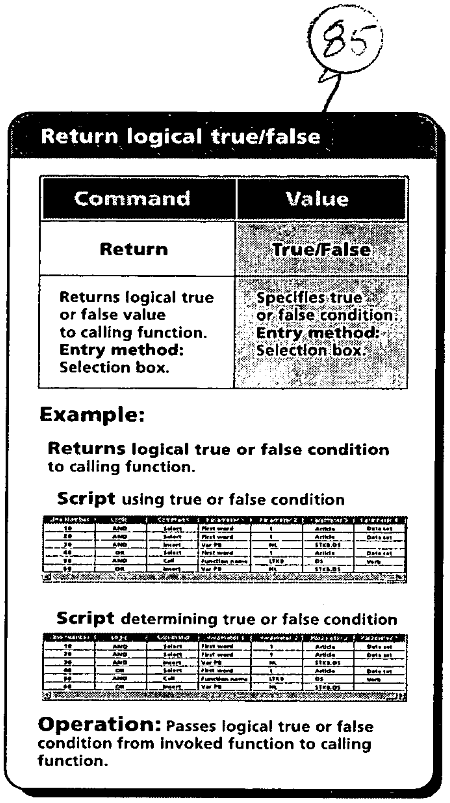
Parameter 1	Parameter 2	Parameter 3	Parameter 4
LTKB/STKB	STR. MEMBER	POINTER	DEFAULT ON/OFF

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Operation: Invokes script associated with PB_x designated by variable name_z. Passes parameters to PB_x script.

Fig. 68

SHEET 70 of 140



Return logical true/false

Command	Value
Return	**True/False**
Returns logical true or false value to calling function. **Entry method:** Selection box.	Specifies true or false condition. **Entry method:** Selection box.

Example:

Returns logical true or false condition to calling function.

Script using true or false condition

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

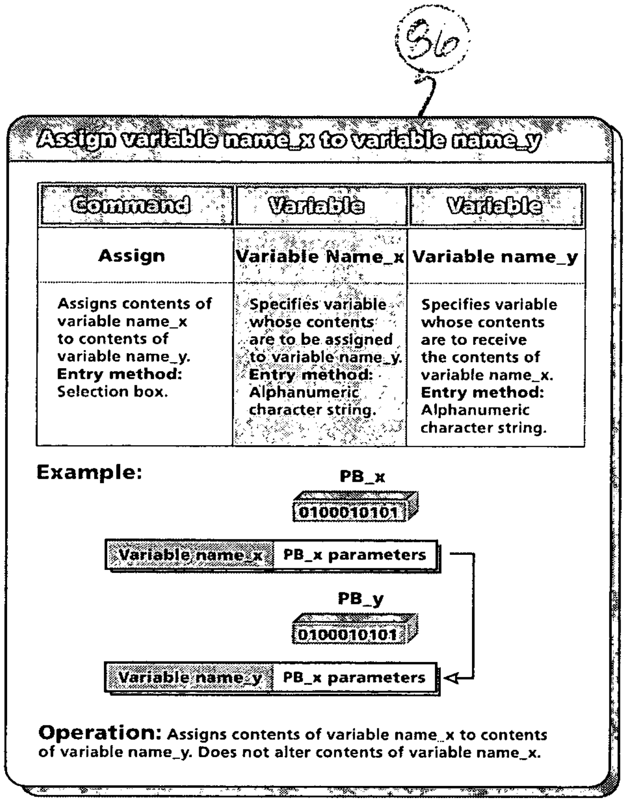
Script determining true or false condition

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Operation: Passes logical true or false condition from invoked function to calling function.

Fig. 69

SHEET 71 OF 140



Assign variable name_x to variable name_y

Command	Variable	Variable
Assign	Variable Name_x	Variable name_y
Assigns contents of variable name_x to contents of variable name_y. **Entry method:** Selection box.	Specifies variable whose contents are to be assigned to variable name_y. **Entry method:** Alphanumeric character string.	Specifies variable whose contents are to receive the contents of variable name_x. **Entry method:** Alphanumeric character string.

Example:

Operation: Assigns contents of variable name_x to contents of variable name_y. Does not alter contents of variable name_x.

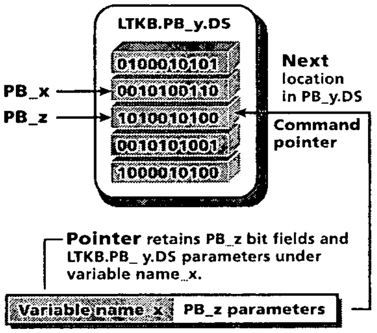
Fig. 70

57

Next PB after PB_x

Command	PB_x
Next	**Variable name_x**
Moves command pointer to next PB (PB_z) after PB_x. **Entry method:** Selection box.	Specifies variable name_x, which contains PB_x. **Entry method:** Alphanumeric character string.

Example:

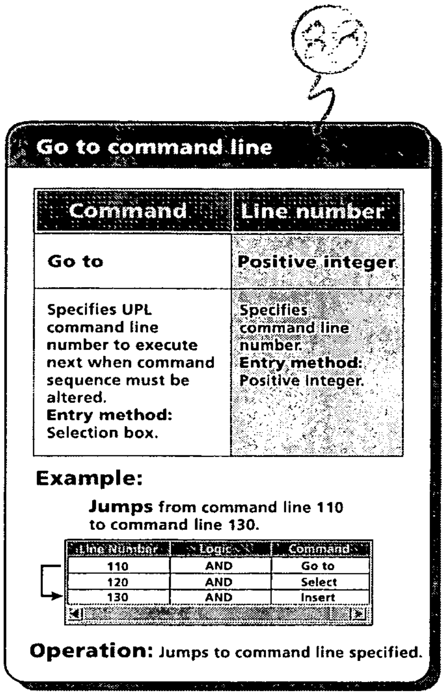


Pointer retains PB_z bit fields and LTKB.PB_ y.DS parameters under variable name_x.

Operation: Transfers contents of variable name_x to PB_z, which is the PB immediately following PB_x in LTKB.PB_y.DS.

Fig. 71

SHEET 73 OF 140



Go to command line

Command	Line number
Go to	**Positive integer**
Specifies UPL command line number to execute next when command sequence must be altered. **Entry method:** Selection box.	Specifies command line number. **Entry method:** Positive Integer.

Example:

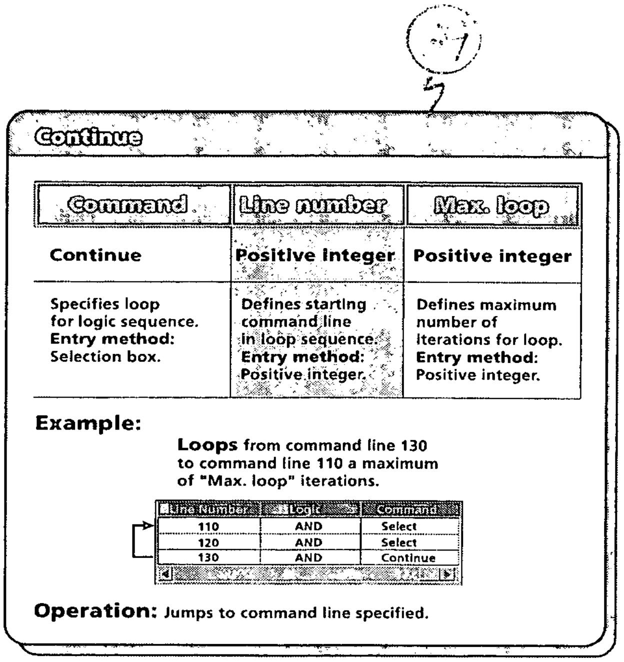
Jumps from command line 110 to command line 130.

Line Number	Logic	Command
110	AND	Go to
120	AND	Select
130	AND	Insert

Operation: Jumps to command line specified.

Fig. 72

SHEET 74 OF 140



Continue

Command	Line number	Max. loop
Continue	**Positive integer**	**Positive integer**
Specifies loop for logic sequence. **Entry method:** Selection box.	Defines starting command line in loop sequence. **Entry method:** Positive integer.	Defines maximum number of iterations for loop. **Entry method:** Positive integer.

Example:

Loops from command line 130 to command line 110 a maximum of "Max. loop" iterations.

Line Number	Logic	Command
110	AND	Select
120	AND	Select
130	AND	Continue

Operation: Jumps to command line specified.

Fig. 73

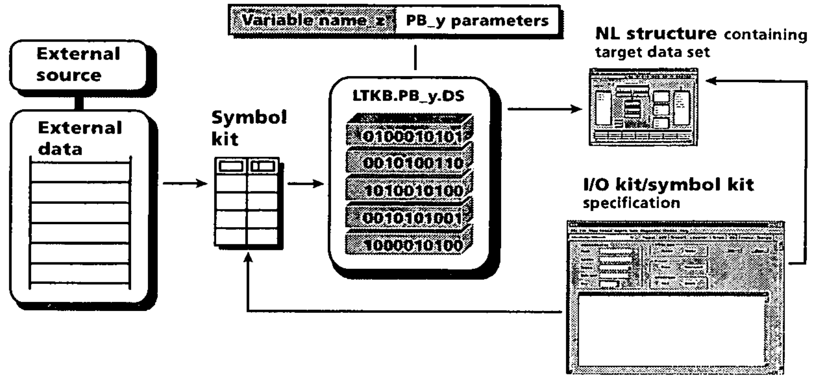
~~90~~ 91

SHEET 75 OF 140

Read (Write) variable name_z from (into) Source

Command	Variable	I/O kit	Symbol kit	Source
Read (Write)	**Variable name_z (Var-PB;Var-DS)**	**Name**	**Name**	**Name**
Reads from or writes to external source and installs or transmits PBs into or from KB structure using specified I/O kit and symbol kit. **Entry method:** Selection box.	Specifies KB structure PB_y into which converted PBs are installed or written to from external device. **Entry method:** Alphanumeric character string.	Defines name of I/O kit used to specify method of I/O for Read or Write command. **Entry method:** Alphanumeric character string.	Defines name of Symbol kit used to specify method of translation between external data structures and PBs. **Entry method:** Alphanumeric character string.	Specifies external machine read from or written to. **Entry method:** Alphanumeric character string.

Example:



Operation: Executes Read or Write from UPL function command line sequence. Uses symbol kit and I/O kit to convert external data structures into PBs while maintaining external machine compatibility. Translation cards and GSM are required for hardware-level integration.

Fig. 74

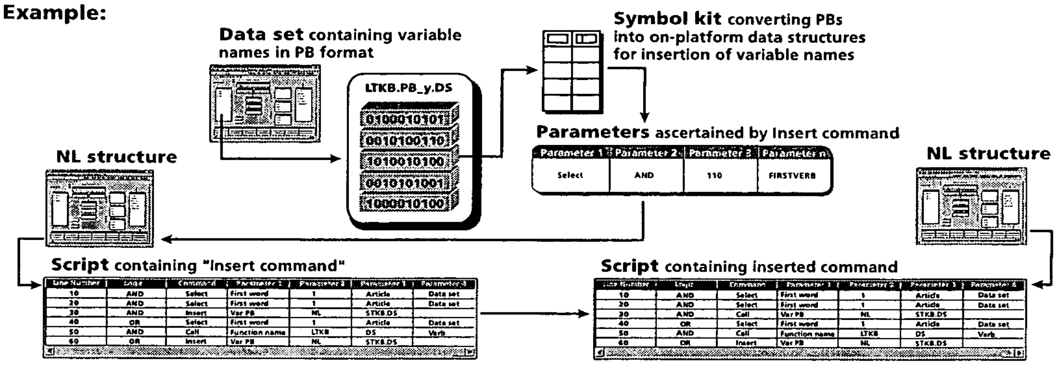
93 92

SHEET 76 OF 140

Insert UPL command

Command	Function PB_y	Parameter 1	Parameter 2	Parameter 3	"n" Parameter
Insert command	Variable name_z	Variable name	Variable name	Variable name	Variable name
Inserts specified UPL command at line number indicated. **Entry method:** Selection box.	Specifies KB structure PB_y containing UPL function into which new UPL command is inserted. **Entry method:** Alphanumeric character string.	Specifies UPL command name (pneumonic) to be inserted. **Entry method:** Alphanumeric character string (generated by KB).	Specifies AND/OR command logic. **Entry method:** Alphanumeric character string (generated by KB).	Specifies UPL command line number. **Entry method:** Alphanumeric character string (generated by KB).	Specifies series of parameters defining specific command operand. **Entry method:** Alphanumeric character strings (generated by KB).

Example:



Data set containing variable names in PB format

Symbol kit converting PBs into on-platform data structures for insertion of variable names

Parameters ascertained by Insert command

Parameter 1	Parameter 2	Parameter 3	Parameter n
Select	AND	110	FIRSTVERB

Script containing "Insert command"

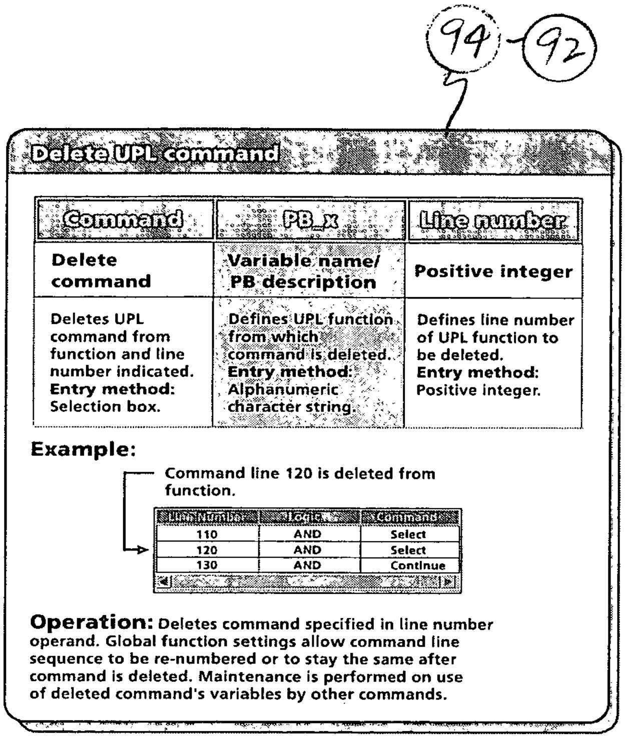
Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Script containing inserted command

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Call	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Operation: Even though the "Insert command" UPL command is indeed a command, it behaves as a UPL function. The function assembles the contents of data sets constructed prior to invoking Insert command action and converts the PBs of the DSs into on-platform structures for use in the targeted script command line. The data sets contain the command mnemonics and operands for the given command inserted. Once Insert command has obtained all parameters required to specify syntax of command, it loads target script with actual command line, including line number, command logic, command name, and related operands. The Insert command operands are "hidden" from the developer at the KDE. Developer enters Insert command and line number only.

Fig. 75



Delete UPL command

Command	PB_x	Line number
Delete command	**Variable name/ PB description**	**Positive integer**
Deletes UPL command from function and line number indicated. **Entry method:** Selection box.	Defines UPL function from which command is deleted. **Entry method:** Alphanumeric character string.	Defines line number of UPL function to be deleted. **Entry method:** Positive integer.

Example:

Command line 120 is deleted from function.

Line Number	Logic	Command
110	AND	Select
120	AND	Select
130	AND	Continue

Operation: Deletes command specified in line number operand. Global function settings allow command line sequence to be re-numbered or to stay the same after command is deleted. Maintenance is performed on use of deleted command's variables by other commands.

Fig. 76

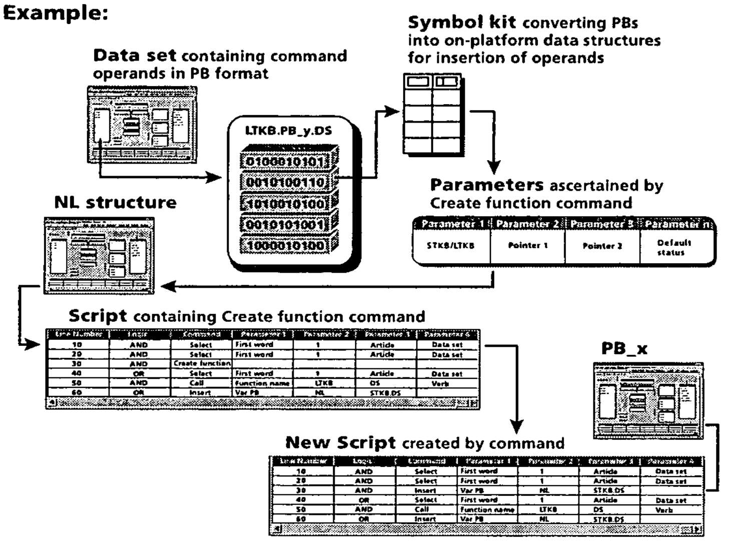
95—92

SHEET 78 OF 140

Create UPL function

Command	PB_x	On-platform	Parameters
Create function	**PB description/ Variable name**	**Variable name**	**P1 through Pn**
Creates UPL function under PB_x. **Entry method:** Selection box.	Node or GF structure containing new function as a structure member. **Entry method:** Alphanumeric character string/ PB bit field Set command.	Defines on-platform character string designating UPL function in Host processor environment. **Entry method:** Alphanumeric character string.	Specifies array of parameters used for KB passed to new function when invoked, and parameters returned to calling function. **Entry method:** Alphanumeric character strings.

Example:



Data set containing command operands in PB format
Symbol kit converting PBs into on-platform data structures for insertion of operands
LTKB.PB_y.DS
0100001010
0010100110
1010010100
0010101001
1000010100
Parameters ascertained by Create function command
NL structure
Script containing Create function command
PB_x
New Script created by command

Parameter 1	Parameter 2	Parameter 3	Parameter n
STKB/LTKB	Pointer 1	Pointer 2	Default status

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Create function				
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Operation: Creates UPL function linked to PB_x. All operands except command name, line number, and sequence logic are hidden from developer. Previously executed parsing and translating logic determines command operands instead of developer's interaction with Script editor. The Create function command assembles the contents of data sets constructed prior to invoking of command and converts the PBs of the DSs into on-platform structures used for generation of new function. The data sets contain the command mnemonics and operands for the new function. Once Create function command has obtained all parameters required to specify syntax of command, it creates new function using specified parameters.

Fig. 77

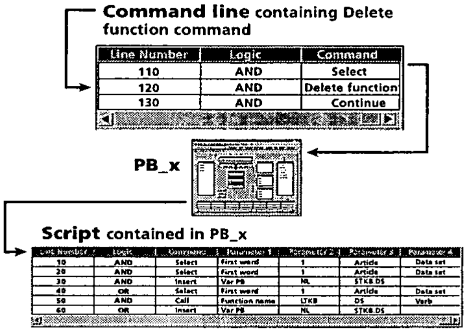
96-92

SHEET 79 OF 140

Delete UPL function

Command	PB_x	Maintenance
Delete function	**Variable name/ PB description**	**Check/ No check**
Deletes UPL function from Node or GF structure. **Entry method:** Selection box.	Defines UPL function to be deleted. **Entry method:** Alphanumeric character string.	Determines whether KDE maintenance is to be performed. **Entry method:** Alphanumeric character string.

Example:



Command line containing Delete function command

Line Number	Logic	Command
110	AND	Select
120	AND	Delete function
130	AND	Continue

Script contained in PB_x

Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
10	AND	Select	First word	1	Article	Data set
20	AND	Select	First word	1	Article	Data set
30	AND	Insert	Var PB	NL	STKB.DS	
40	OR	Select	First word	1	Article	Data set
50	AND	Call	Function name	LTKB	DS	Verb
60	OR	Insert	Var PB	NL	STKB.DS	

Operation: Deletes UPL function specified by PB_x and optionally performs KB maintenance.

Fig. 78

SHEET 80 OF 140

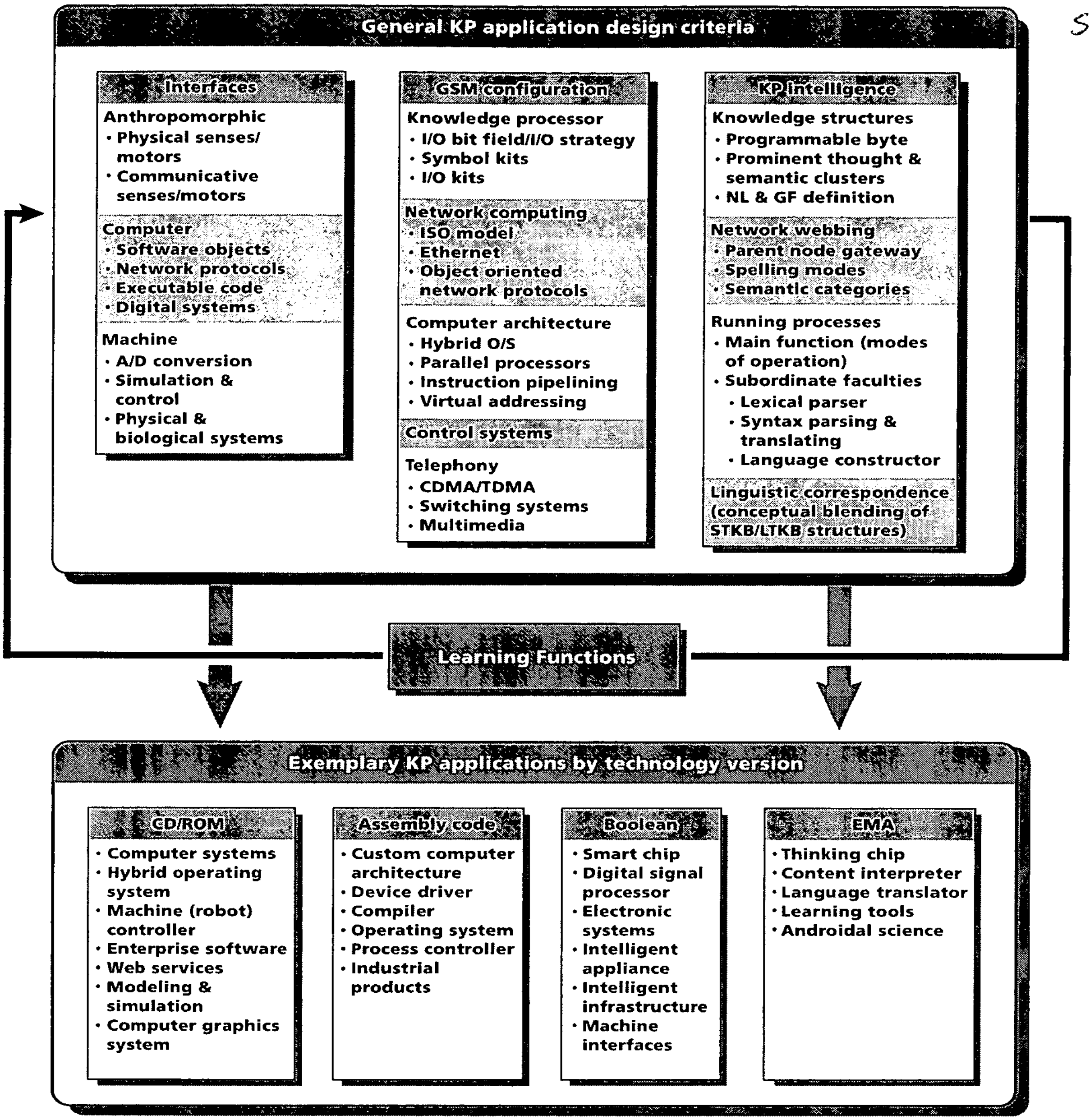
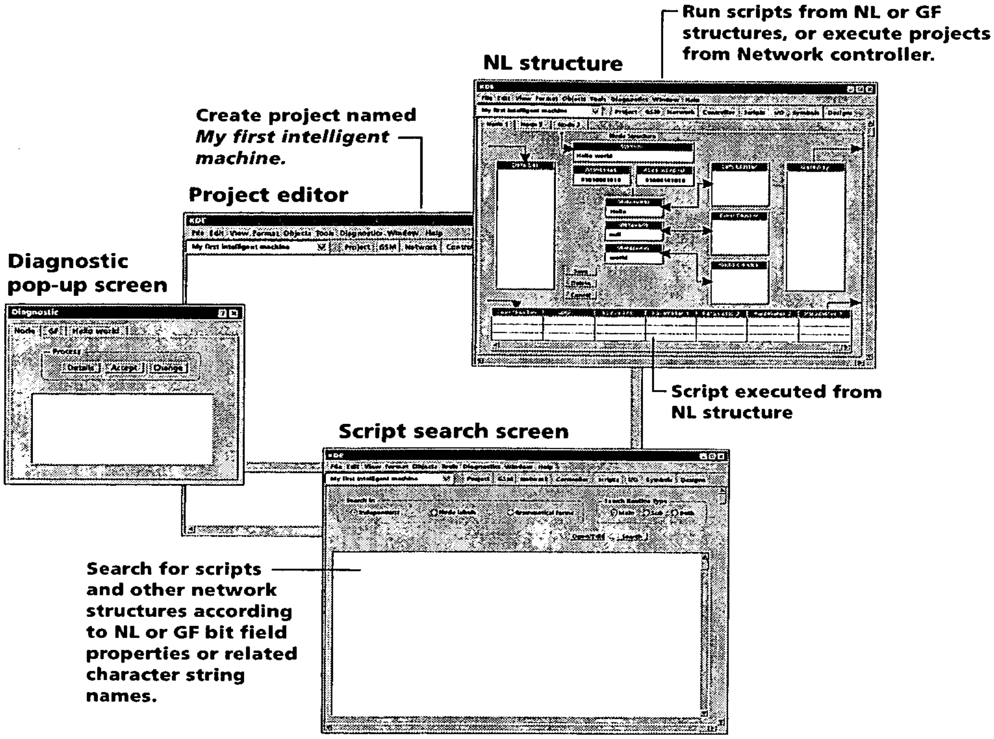


Fig. 79

SHEET 81 OF 140



NL structure

Run scripts from NL or GF structures, or execute projects from Network controller.

Create project named *My first intelligent machine.*

Project editor

Diagnostic pop-up screen

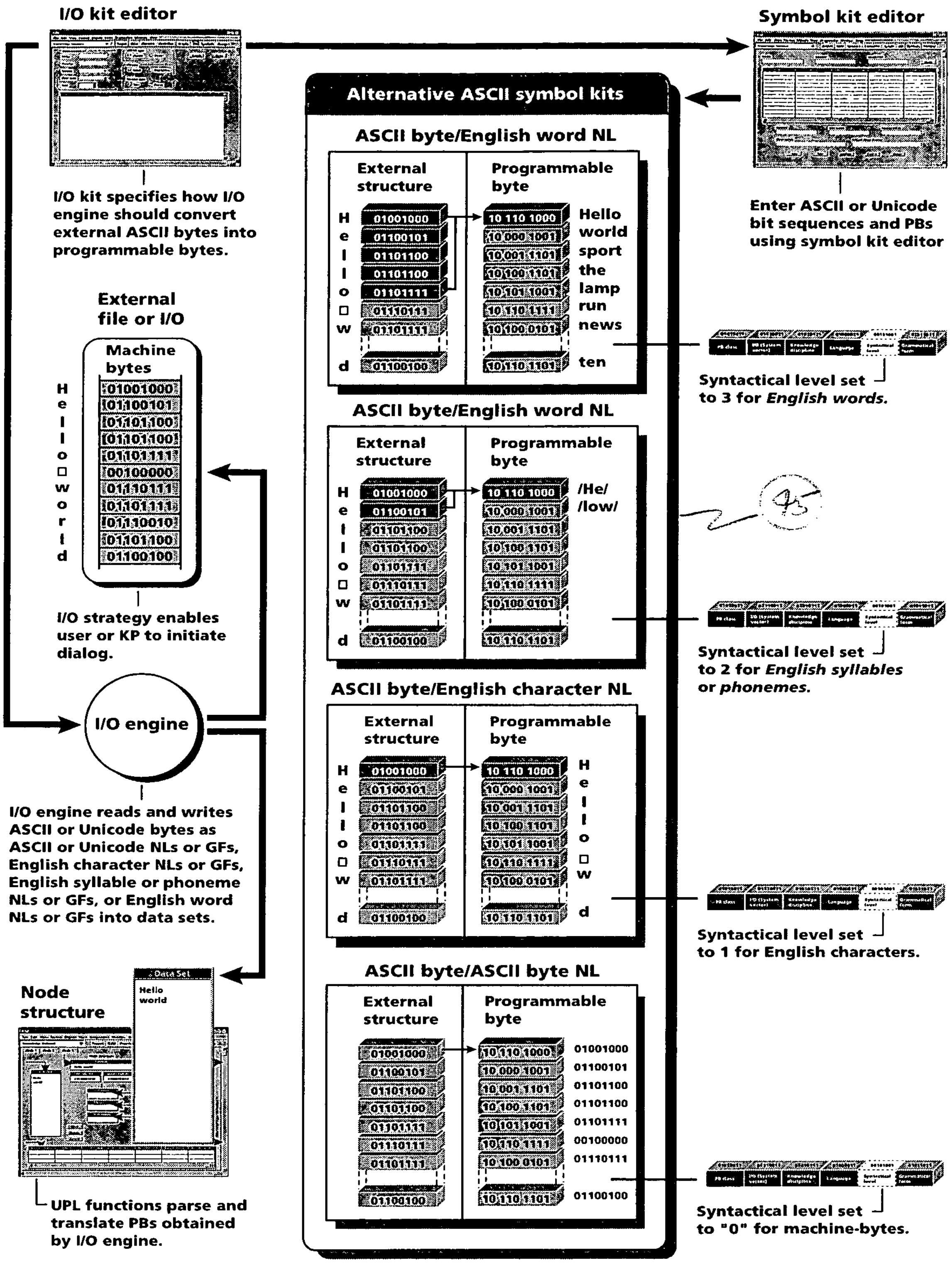
Script executed from NL structure

Script search screen

Search for scripts and other network structures according to NL or GF bit field properties or related character string names.

Fig. 80

SHEET 52 OF 140

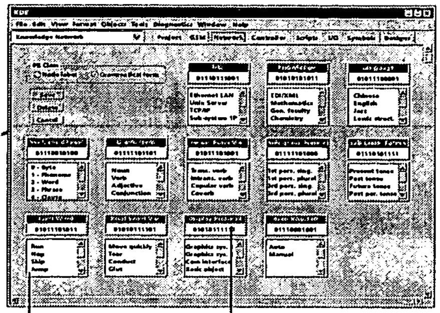


I/O kit editor
I/O kit specifies how I/O engine should convert external ASCII bytes into programmable bytes.
Symbol kit editor
Enter ASCII or Unicode bit sequences and PBs using symbol kit editor
Alternative ASCII symbol kits
ASCII byte/English word NL
External structure
Programmable byte
H
e
l
l
o
□
w
d
01001000
01100101
01101100
01101100
01101111
01110111
01101111
01100100
10 110 1000
10 000 1001
10 001 1101
10 100 1101
10 101 1001
10 110 1111
10 100 0101
10 110 1101
Hello
world
sport
the
lamp
run
news
ten
Syntactical level set to 3 for English words.
ASCII byte/English word NL
External structure
Programmable byte
/He/
/low/
Syntactical level set to 2 for English syllables or phonemes.
45
ASCII byte/English character NL
External structure
Programmable byte
H
e
l
l
o
□
w
d
Syntactical level set to 1 for English characters.
ASCII byte/ASCII byte NL
External structure
Programmable byte
01001000
01100101
01101100
01101100
01101111
00100000
01110111
01100100
Syntactical level set to "0" for machine-bytes.
External file or I/O
Machine bytes
H
e
l
l
o
□
w
o
r
l
d
01001000
01100101
01101100
01101100
01101111
00100000
01110111
01101111
01110010
01101100
01100100
I/O strategy enables user or KP to initiate dialog.
I/O engine
I/O engine reads and writes ASCII or Unicode bytes as ASCII or Unicode NLs or GFs, English character NLs or GFs, English syllable or phoneme NLs or GFs, or English word NLs or GFs into data sets.
Node structure
Data Set
Hello
world
UPL functions parse and translate PBs obtained by I/O engine.

Fig. 81

SHEET 83 OF 140

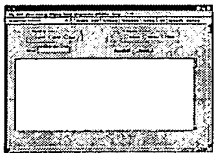
PB settings editor



Enter PB settings options

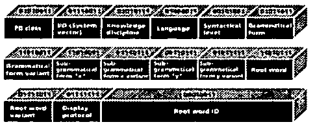
Select PB bit fields for actual PB settings

PB search screen



Create or alter PBs by using PB settings editor and PB search screen

PB bit field definitions for *Hello world* project



I/O engine serves Host and Internet applications.

Non-specific intellectual faculty for demonstration

Arbitrary languages for demonstration

Syntactical levels from 0 for byte to 8 for text, including levels 5 (clause), 6 (sentence), 7 (paragraph), and 8 (text format)

gf and gf variants for English, Spanish, Arithmetic, EDI, and XML

I/O	Knowledge	Language	Syntact level	Gram. form	GF Variant
Host file	Gen/Demo	English	0-byte	Noun	
Keyboard		Spanish	1-character	Adjective	Transitive *
Monitor		Arithmetic	2-phoneme	Verb	Intransitive
Basic object		EDI	3-word	Article	Copular
Net. protocol		XML	4-phrase	Adverb	

*Shaded areas represent hierarchical relationships (i.e., verb/type of verb)

Matrix combinations for sub-grammatical forms

Semantic encodings

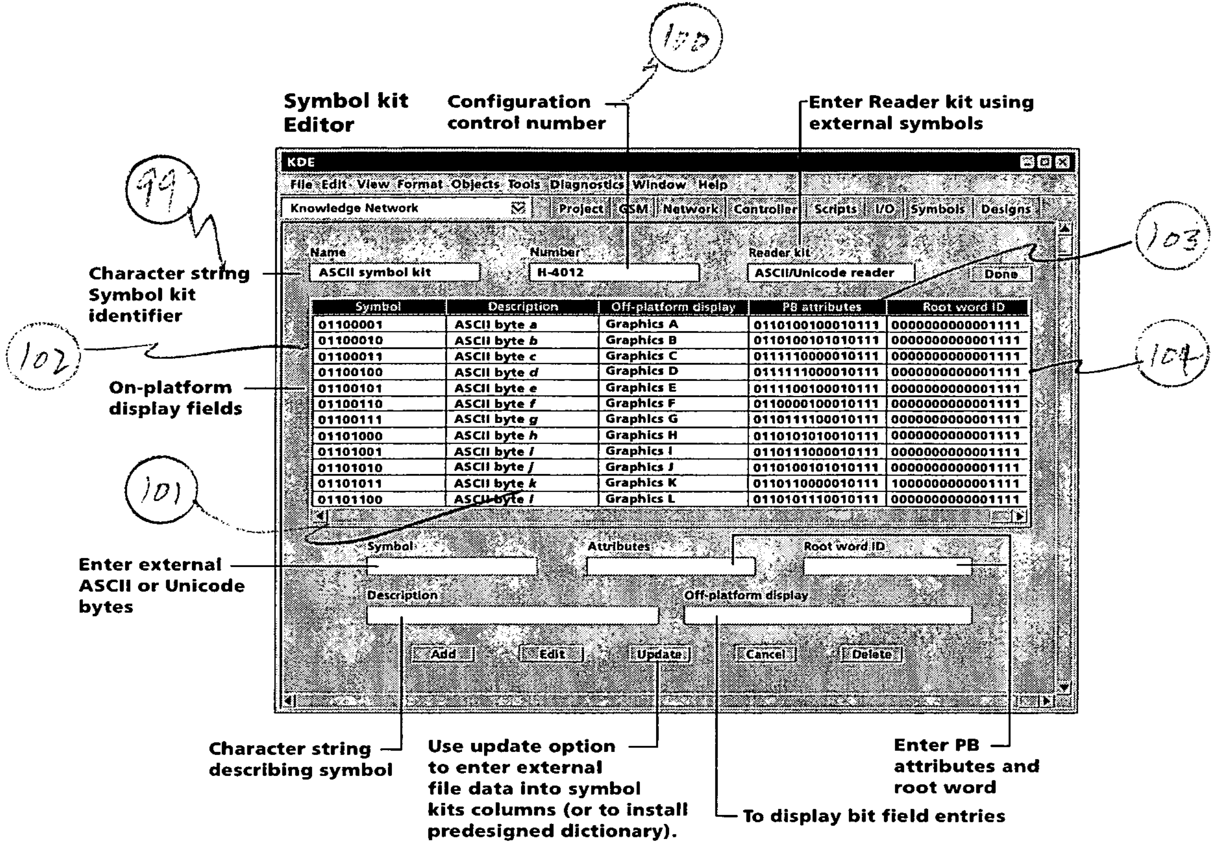
Off-platform display

Automatic generation of RW ID

SGF x	SGF y	Root word	RW variant	Display	RW ID
1st pers. sing.	Present tense	world *	Universe	Graphic arts	Auto
1st pers. plur.	Past tense	Hello	Scope		
3rd pers. sing.	Future tense	2	globe		
3rd pers. plur.	Past perfect	+	Personality		
2nd person	Future perfect	=	humanity		

*Shaded areas represent hierarchical relationships (i.e., root word/sense)

Fig. 82



100
99
102
103
104
101
KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
Done
Add
Edit
Update
Cancel
Delete

Symbol kit Editor

Configuration control number

Enter Reader kit using external symbols

Character string Symbol kit identifier

On-platform display fields

Enter external ASCII or Unicode bytes

Name: ASCII symbol kit

Number: H-4012

Reader kit: ASCII/Unicode reader

Symbol	Description	Off-platform display	PB attributes	Root word ID
01100001	ASCII byte *a*	Graphics A	0110100100010111	0000000000001111
01100010	ASCII byte *b*	Graphics B	0110100101010111	0000000000001111
01100011	ASCII byte *c*	Graphics C	0111110000010111	0000000000001111
01100100	ASCII byte *d*	Graphics D	0111111000010111	0000000000001111
01100101	ASCII byte *e*	Graphics E	0111100100010111	0000000000001111
01100110	ASCII byte *f*	Graphics F	0110000100010111	0000000000001111
01100111	ASCII byte *g*	Graphics G	0110111100010111	0000000000001111
01101000	ASCII byte *h*	Graphics H	0110101010010111	0000000000001111
01101001	ASCII byte *i*	Graphics I	0110111000010111	0000000000001111
01101010	ASCII byte *j*	Graphics J	0110100101010111	0000000000001111
01101011	ASCII byte *k*	Graphics K	0110110000010111	1000000000001111
01101100	ASCII byte *l*	Graphics L	0110101110010111	0000000000001111

Symbol:

Attributes:

Root word ID:

Description:

Off-platform display:

Character string describing symbol

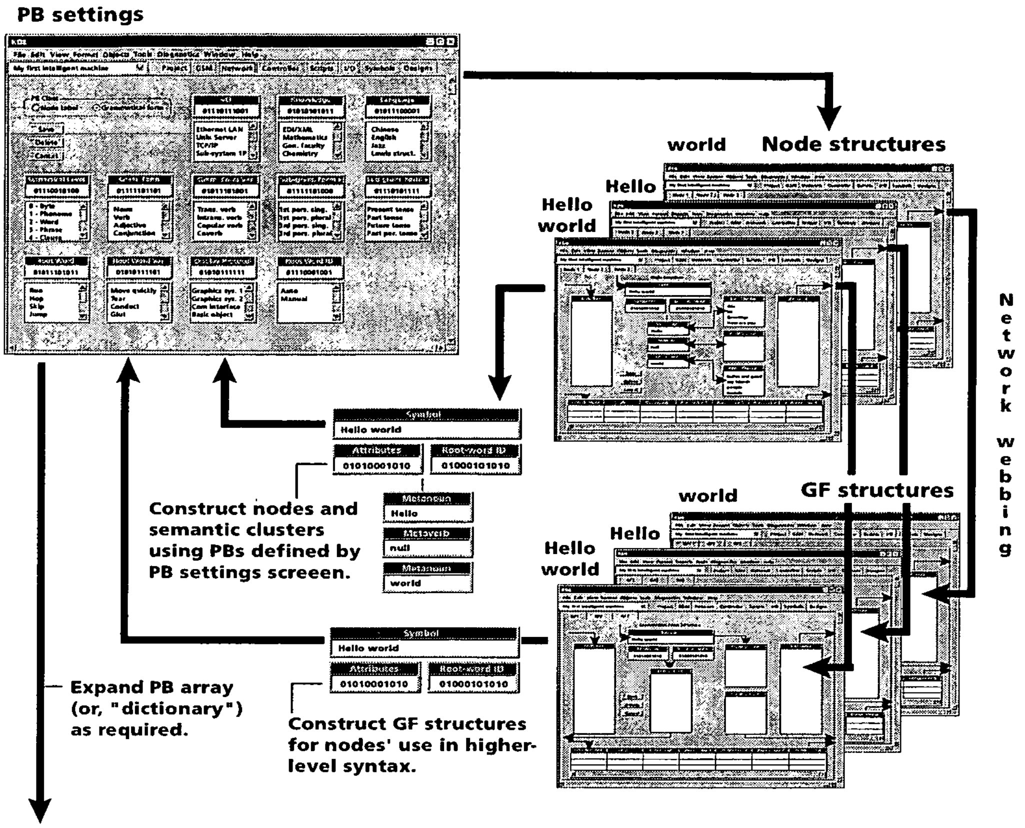
Use update option to enter external file data into symbol kits columns (or to install predesigned dictionary).

Enter PB attributes and root word

To display bit field entries

Fig. 83

SHEET 55 OF 140



100 most commonly used English words (Augmented by *Hello world* project's partial vocabulary)

the
of
and
a
to
in
is
you
that
it
he
for
was
on
are
but
what
all

were
when
we
there
can
an
your
which
their
said
if
do
into
has
more
her
two
like

him
see
time
could
no
make
than
first
been
long
little
very
after
words
called
just
where
most

know
get
through
back
much
before
also
around
another
came
come
work
three
word
must
because
does
part

even
place
well
as
with
his
they
at
be
this
from
I
have
or
by
one
had
not

will
each
about
how
up
out
them
then
she
many
some
so
these
would
other
its
who
now

people
my
made
over
did
down
only
way
find
use
may
water
go
good
new
write
our
used

me
man
too
any
day
same
right
look
think
such
here
take
why
things
help
put
years
different

away
again
off
went
old
number
how
why
where
when
what
2
+
4
fact
hello
world
Andrew

Fig. 84

(119)

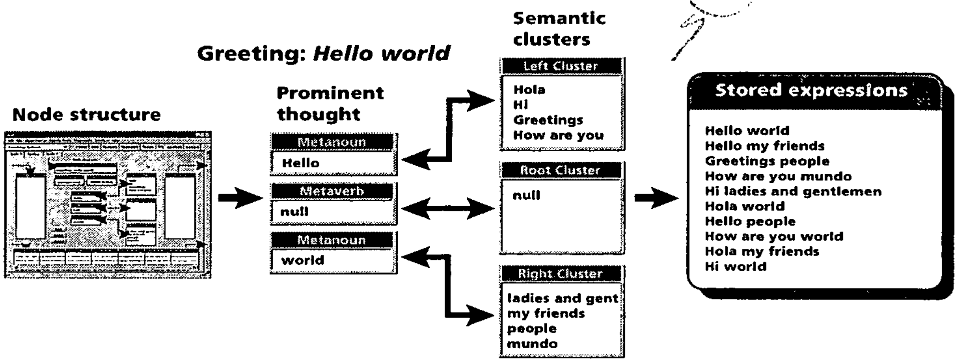

Greeting: *Hello world*
Node structure
Prominent thought
Metanoun
Hello
Metaverb
null
Metanoun
world
Semantic clusters
Left Cluster
Hola
Hi
Greetings
How are you
Root Cluster
null
Right Cluster
ladies and gent
my friends
people
mundo
Stored expressions
Hello world
Hello my friends
Greetings people
How are you mundo
Hi ladies and gentlemen
Hola world
Hello people
How are you world
Hola my friends
Hi world

Fig. 85(a)

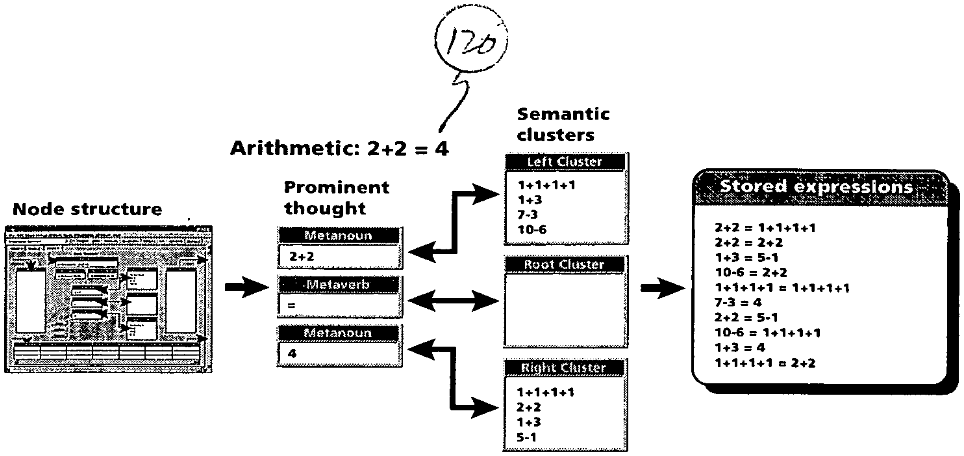
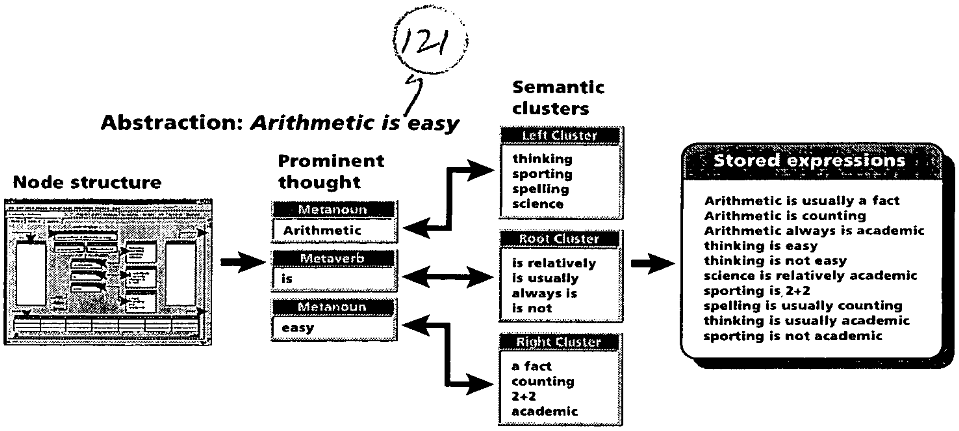

120
Arithmetic: 2+2 = 4
Node structure
Prominent thought
Metanoun
2+2
Metaverb
=
Metanoun
4
Semantic clusters
Left Cluster
1+1+1+1
1+3
7-3
10-6
Root Cluster
Right Cluster
1+1+1+1
2+2
1+3
5-1
Stored expressions
2+2 = 1+1+1+1
2+2 = 2+2
1+3 = 5-1
10-6 = 2+2
1+1+1+1 = 1+1+1+1
7-3 = 4
2+2 = 5-1
10-6 = 1+1+1+1
1+3 = 4
1+1+1+1 = 2+2

Fig. 85(b)



Arithmetic is usually a fact
Arithmetic is counting
Arithmetic always is academic
thinking is easy
thinking is not easy
science is relatively academic
sporting is 2+2
spelling is usually counting
thinking is usually academic
sporting is not academic

Fig. 85(c)

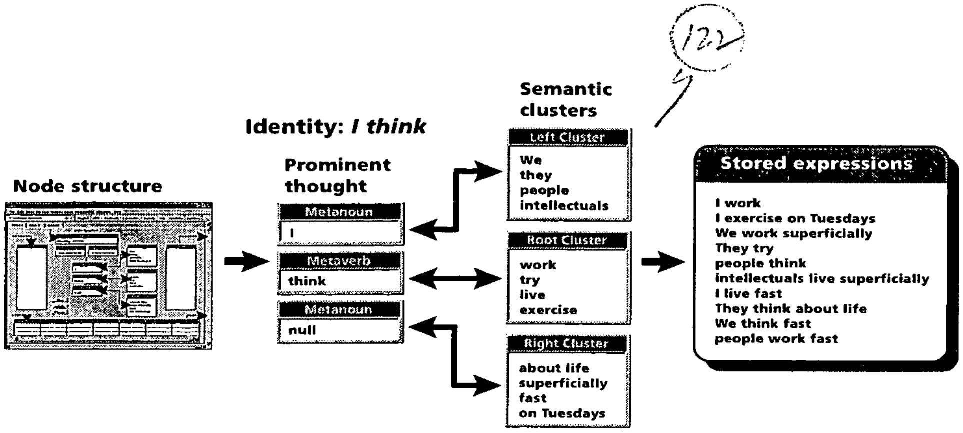

122
Identity: *I think*
Node structure
Prominent thought
Metanoun
I
Metaverb
think
Metanoun
null
Semantic clusters
Left Cluster
We
they
people
intellectuals
Root Cluster
work
try
live
exercise
Right Cluster
about life
superficially
fast
on Tuesdays
Stored expressions
I work
I exercise on Tuesdays
We work superficially
They try
people think
intellectuals live superficially
I live fast
They think about life
We think fast
people work fast

Fig. 85(d)

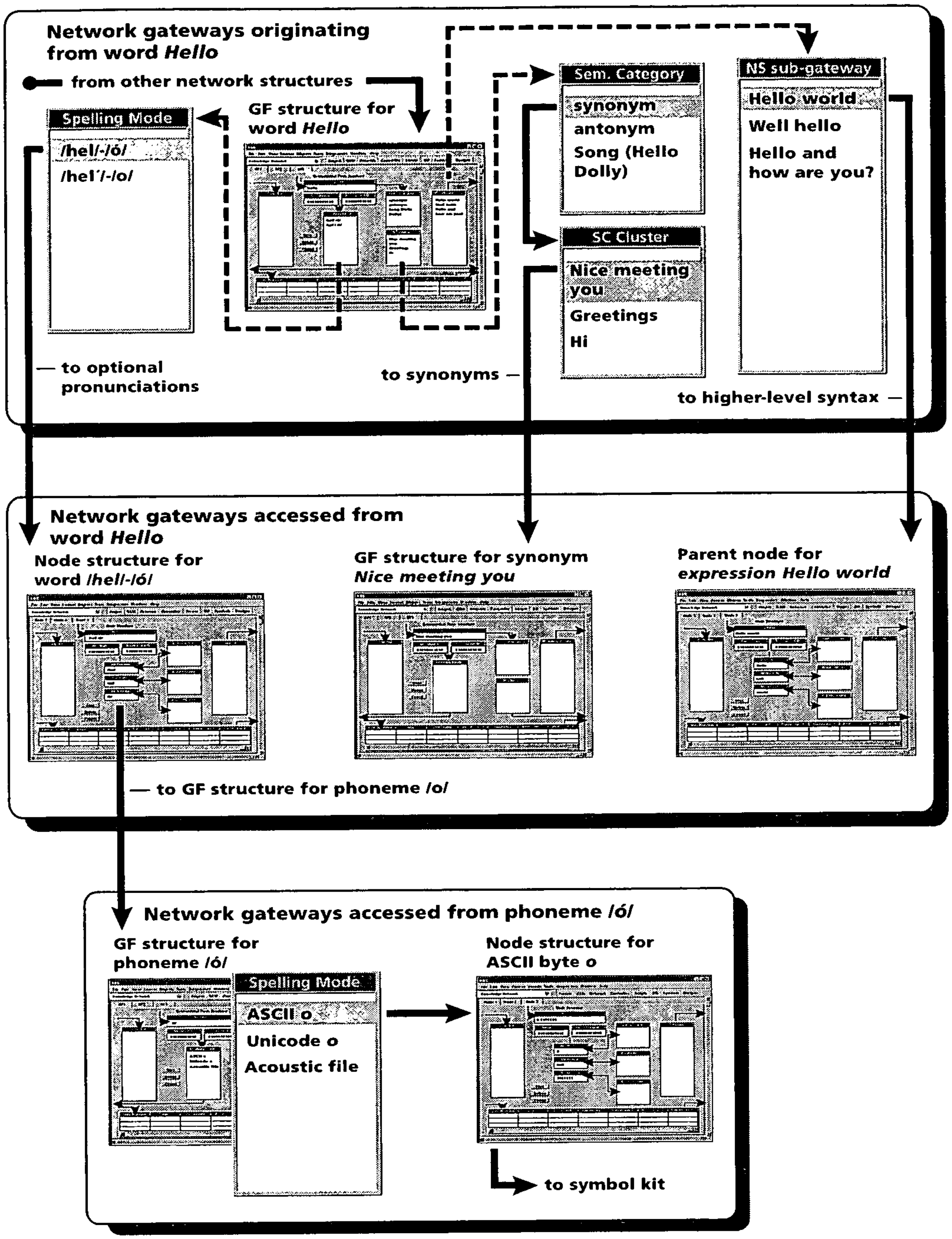
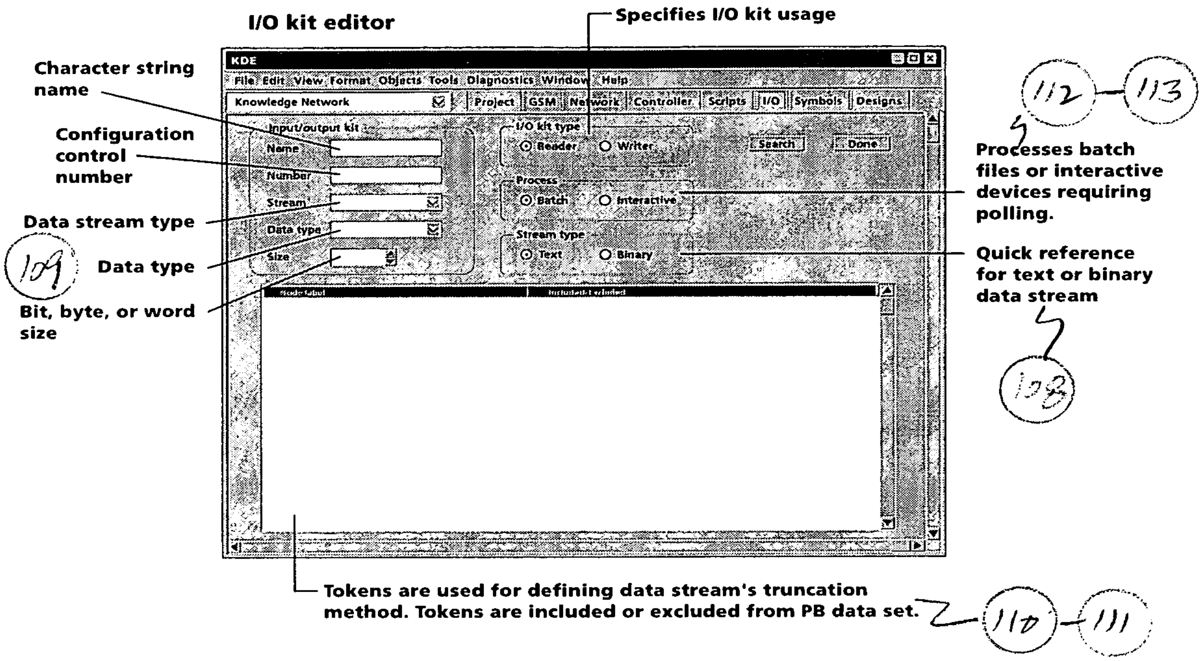

Network gateways originating from word *Hello*
from other network structures
Spelling Mode
/hel/-/ó/
/hel´/-/o/
GF structure for word *Hello*
Sem. Category
synonym
antonym
Song (Hello Dolly)
SC Cluster
Nice meeting you
Greetings
Hi
NS sub-gateway
Hello world
Well hello
Hello and how are you?
to optional pronunciations
to synonyms
to higher-level syntax
Network gateways accessed from word *Hello*
Node structure for word */hel/-/ó/*
GF structure for synonym *Nice meeting you*
Parent node for *expression Hello world*
to GF structure for phoneme /o/
Network gateways accessed from phoneme /ó/
GF structure for phoneme /ó/
Spelling Mode
ASCII o
Unicode *o*
Acoustic file
Node structure for ASCII byte *o*
to symbol kit

Fig. 86


I/O kit editor
Specifies I/O kit usage
KDE
File Edit View Format Objects Tools Diagnostics Window Help
Knowledge Network
Project GSM Network Controller Scripts I/O Symbols Designs
Input/output kit
Name
Number
Stream
Data type
Size
Search
Done
Node label
Included/Excluded
Character string name
Configuration control number
Data stream type
Data type
Bit, byte, or word size
109
112
113
Processes batch files or interactive devices requiring polling.
Quick reference for text or binary data stream
108
Tokens are used for defining data stream's truncation method. Tokens are included or excluded from PB data set.
110
111

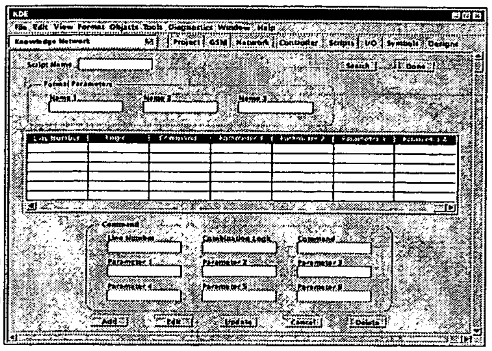
I/O kit type: ⊙ Reader ○ Writer

Process: ⊙ Batch ○ Interactive

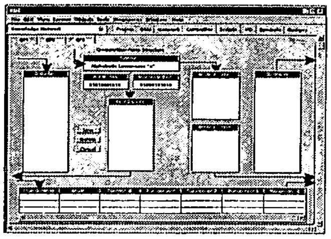
Stream type: ⊙ Text ○ Binary

Fig. 87

Script editor



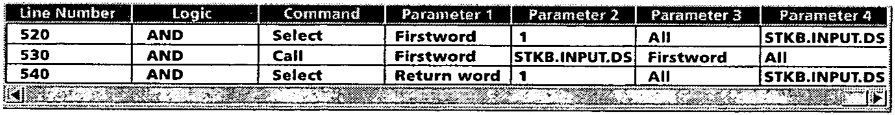
GF structure for *Hello world*



116

Main script is installed into GF structure for *Hello world*

Main script

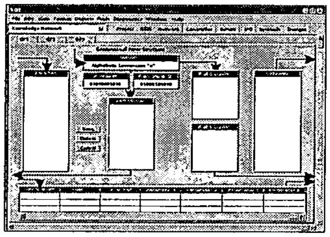


Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
520	AND	Select	Firstword	1	All	STKB.INPUT.DS
530	AND	Call	Firstword	STKB.INPUT.DS	Firstword	All
540	AND	Select	Return word	1	All	STKB.INPUT.DS

I/O strategy is motivated to acquire new knowledge

I/O strategy determines when to create new knowledge internally, when to engage in dialog, and when to initiate subordinate functions.

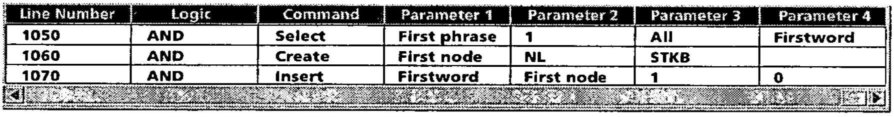
GF structure for word *what*



117

Exemplary Subordinate script analyzes interrogative sentence beginning with word *what*.

Subordinate script



Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
1050	AND	Select	First phrase	1	All	Firstword
1060	AND	Create	First node	NL	STKB	
1070	AND	Insert	Firstword	First node	1	0

Fig. 88

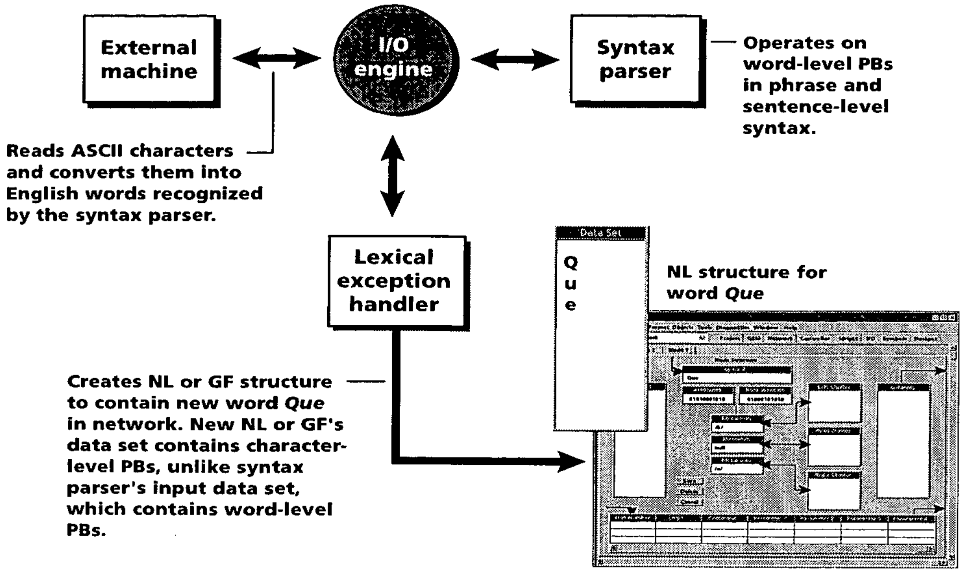

External machine
I/O engine
Syntax parser
Operates on word-level PBs in phrase and sentence-level syntax.
Reads ASCII characters and converts them into English words recognized by the syntax parser.
Lexical exception handler
Data Set
Q
u
e
NL structure for word *Que*
Creates NL or GF structure to contain new word *Que* in network. New NL or GF's data set contains character-level PBs, unlike syntax parser's input data set, which contains word-level PBs.

Fig. 89

SHEET 94 OF 140

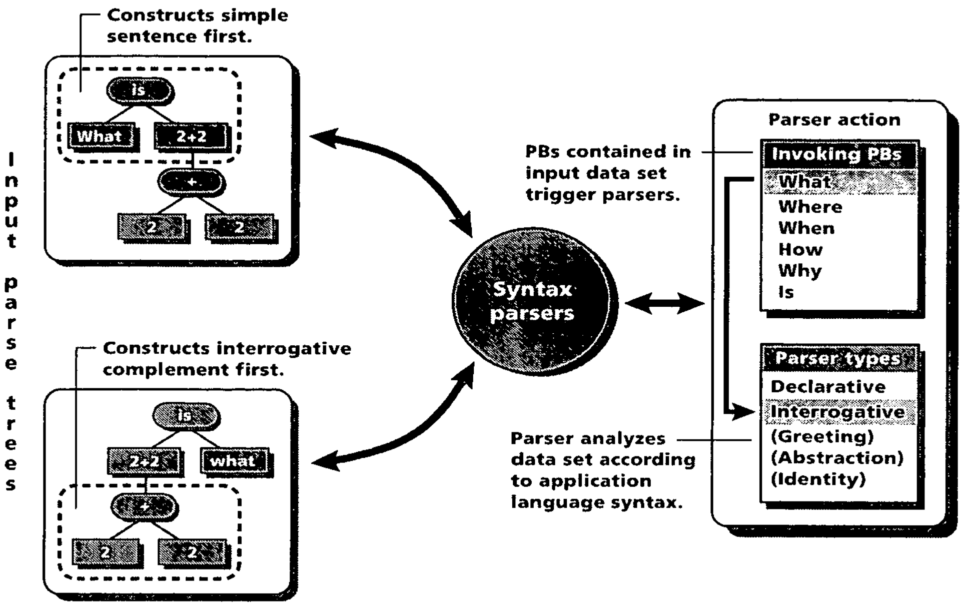

Constructs simple sentence first.
Is
What
2+2
+
2
2
Input parse trees
Constructs interrogative complement first.
Is
2+2
what
+
2
2
Syntax parsers
PBs contained in input data set trigger parsers.
Parser action
Invoking PBs
What
Where
When
How
Why
Is
Parser types
Declarative
Interrogative
(Greeting)
(Abstraction)
(Identity)
Parser analyzes data set according to application language syntax.

Fig. 90

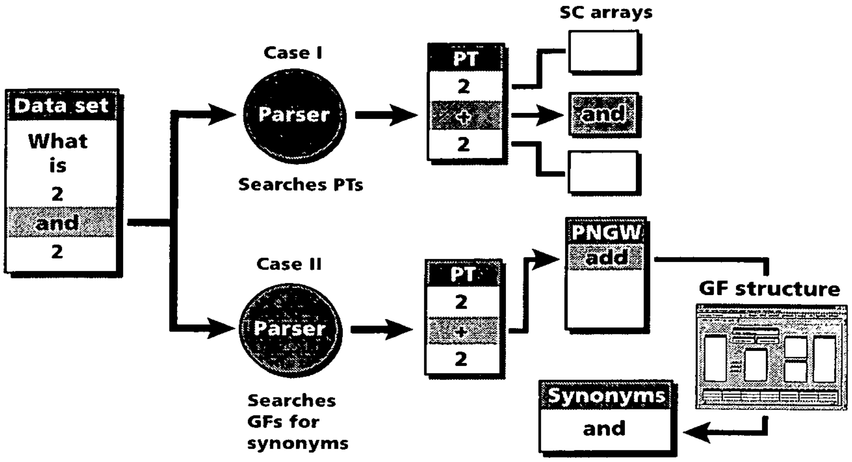
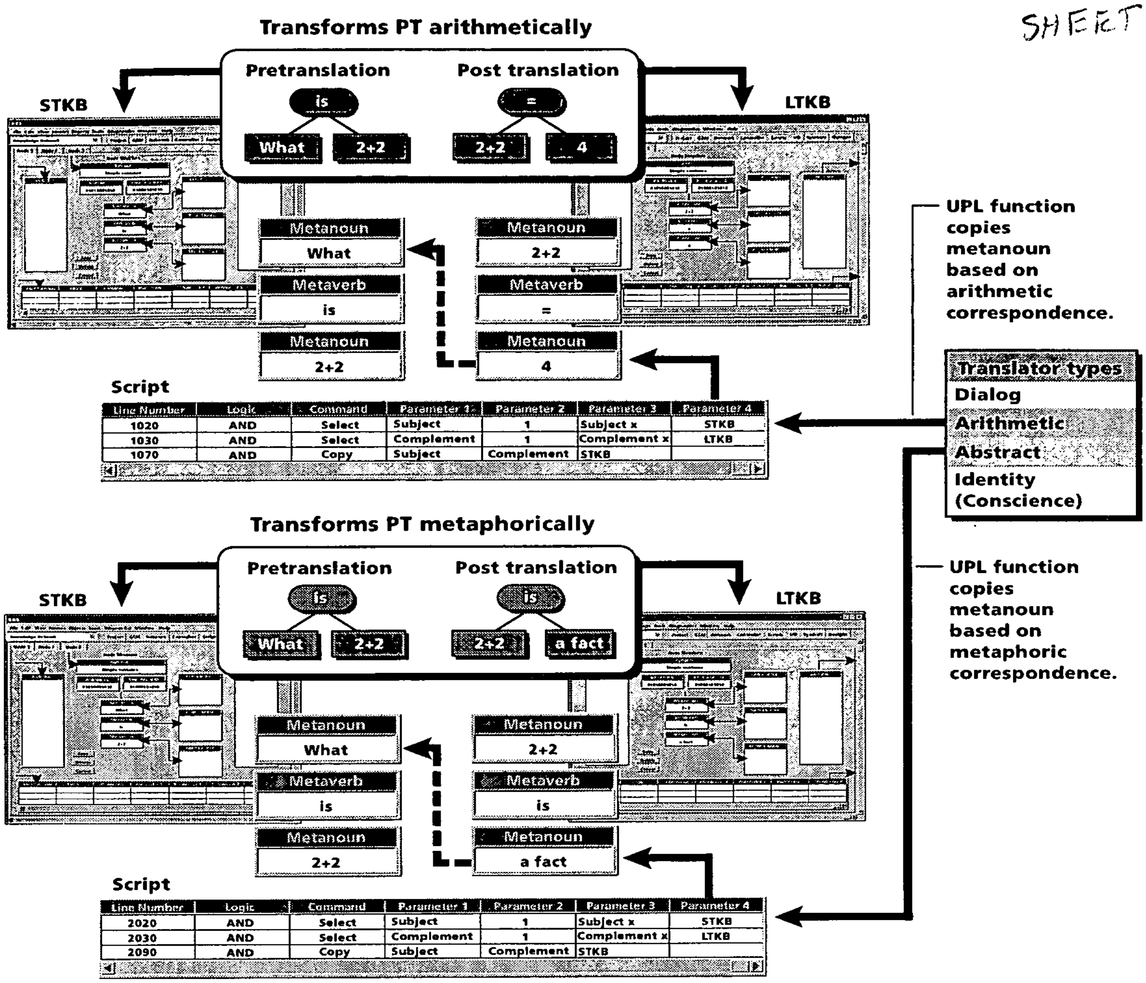

SC arrays
Case I
PT
2
+
2
and
Data set
What
is
2
and
2
Parser
Searches PTs
PNGW
add
Case II
PT
2
+
2
GF structure
Parser
Synonyms
and
Searches
GFs for
synonyms

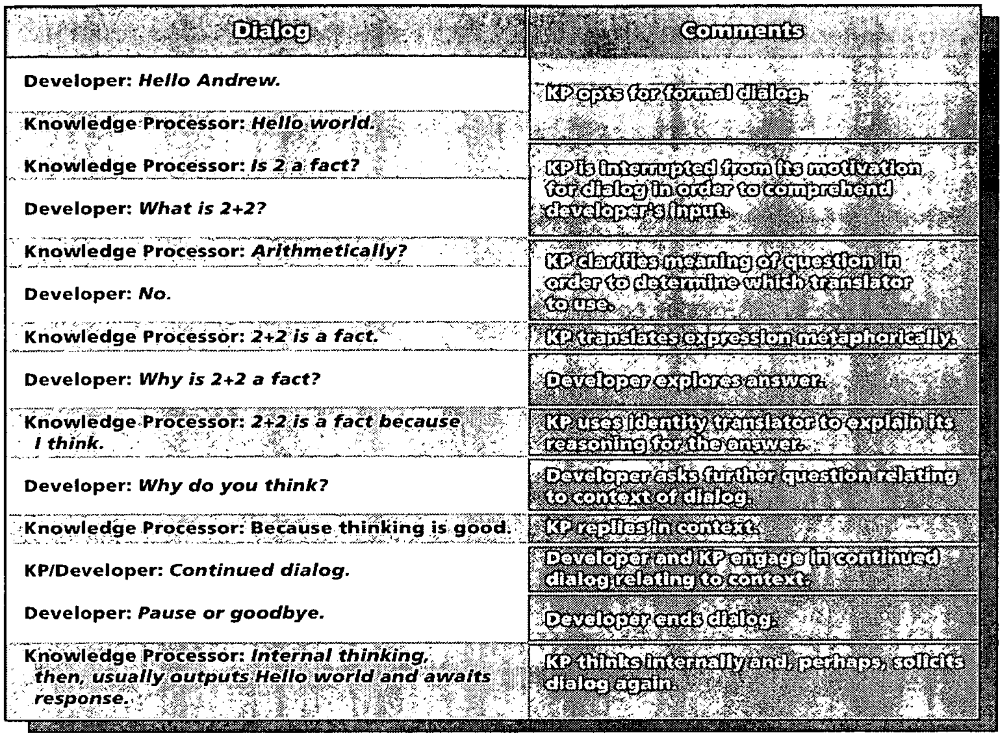
Fig. 91



Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
1020	AND	Select	Subject	1	Subject x	STKB
1030	AND	Select	Complement	1	Complement x	LTKB
1070	AND	Copy	Subject	Complement	STKB	

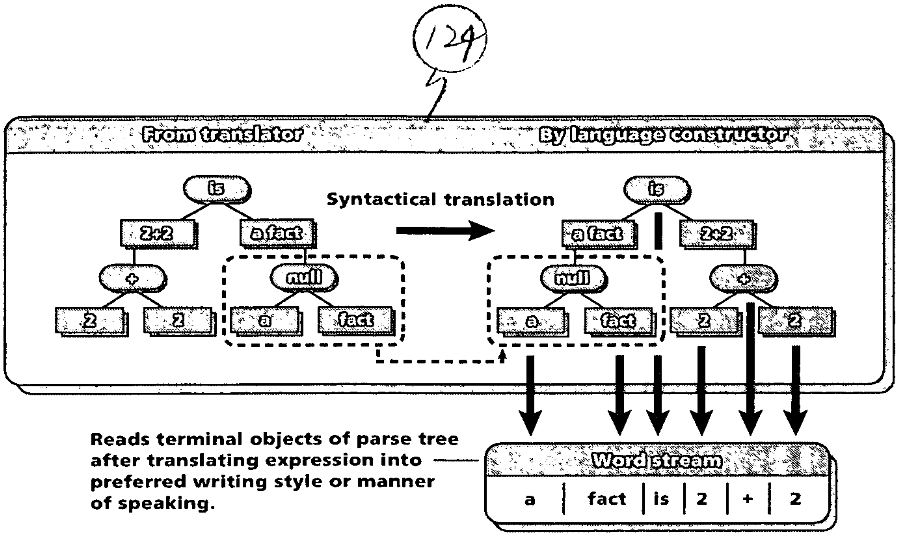
Line Number	Logic	Command	Parameter 1	Parameter 2	Parameter 3	Parameter 4
2020	AND	Select	Subject	1	Subject x	STKB
2030	AND	Select	Complement	1	Complement x	LTKB
2090	AND	Copy	Subject	Complement	STKB	

Fig. 92



Dialog	Comments
Developer: ***Hello Andrew.***	KP opts for formal dialog.
Knowledge Processor: ***Hello world.***	
Knowledge Processor: ***Is 2 a fact?***	KP is interrupted from its motivation for dialog in order to comprehend developer's input.
Developer: ***What is 2+2?***	
Knowledge Processor: ***Arithmetically?***	KP clarifies meaning of question in order to determine which translator to use.
Developer: ***No.***	
Knowledge Processor: ***2+2 is a fact.***	KP translates expression metaphorically.
Developer: ***Why is 2+2 a fact?***	Developer explores answer.
Knowledge Processor: ***2+2 is a fact because I think.***	KP uses identity translator to explain its reasoning for the answer.
Developer: ***Why do you think?***	Developer asks further question relating to context of dialog.
Knowledge Processor: Because thinking is good.	KP replies in context.
KP/Developer: ***Continued dialog.***	Developer and KP engage in continued dialog relating to context.
Developer: ***Pause or goodbye.***	Developer ends dialog.
Knowledge Processor: ***Internal thinking, then, usually outputs Hello world and awaits response.***	KP thinks internally and, perhaps, solicits dialog again.

Fig. 93



Reads terminal objects of parse tree after translating expression into preferred writing style or manner of speaking.

Fig. 94

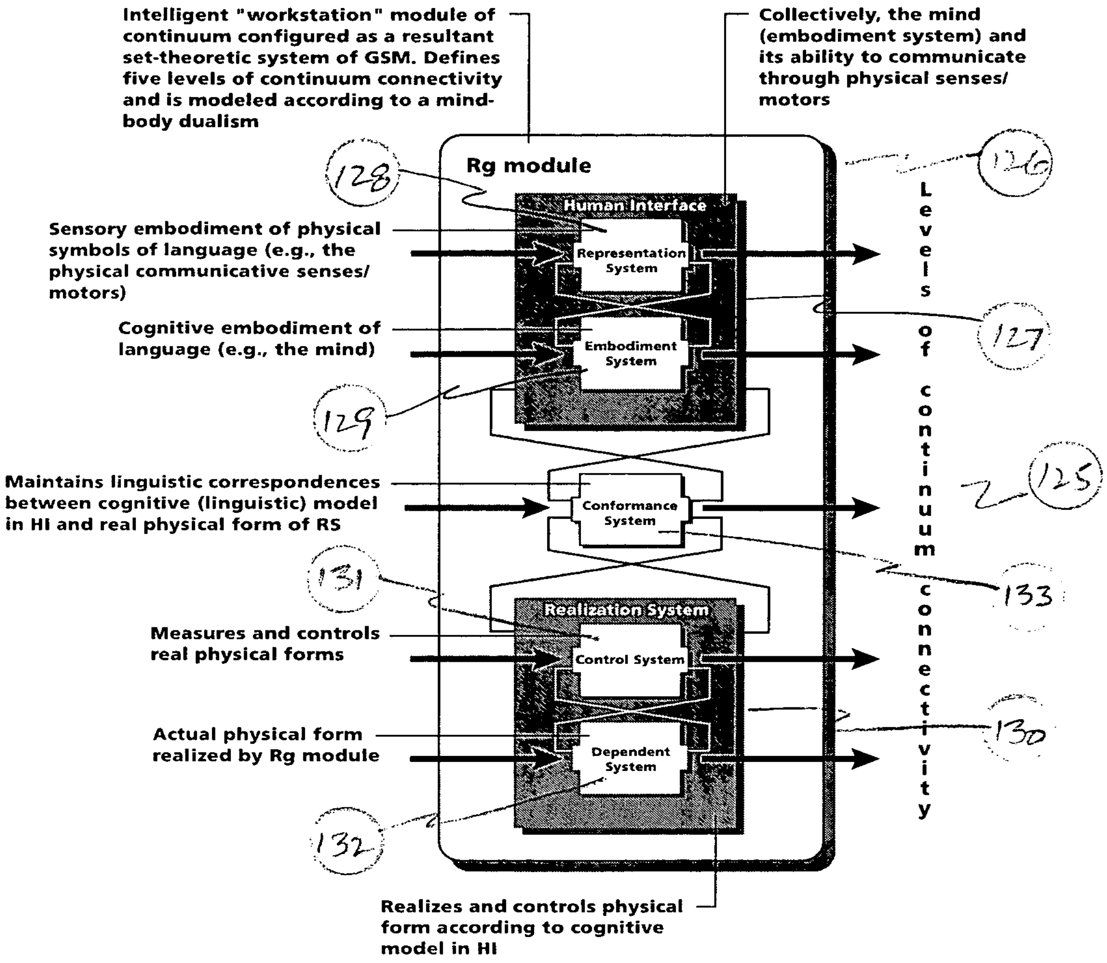
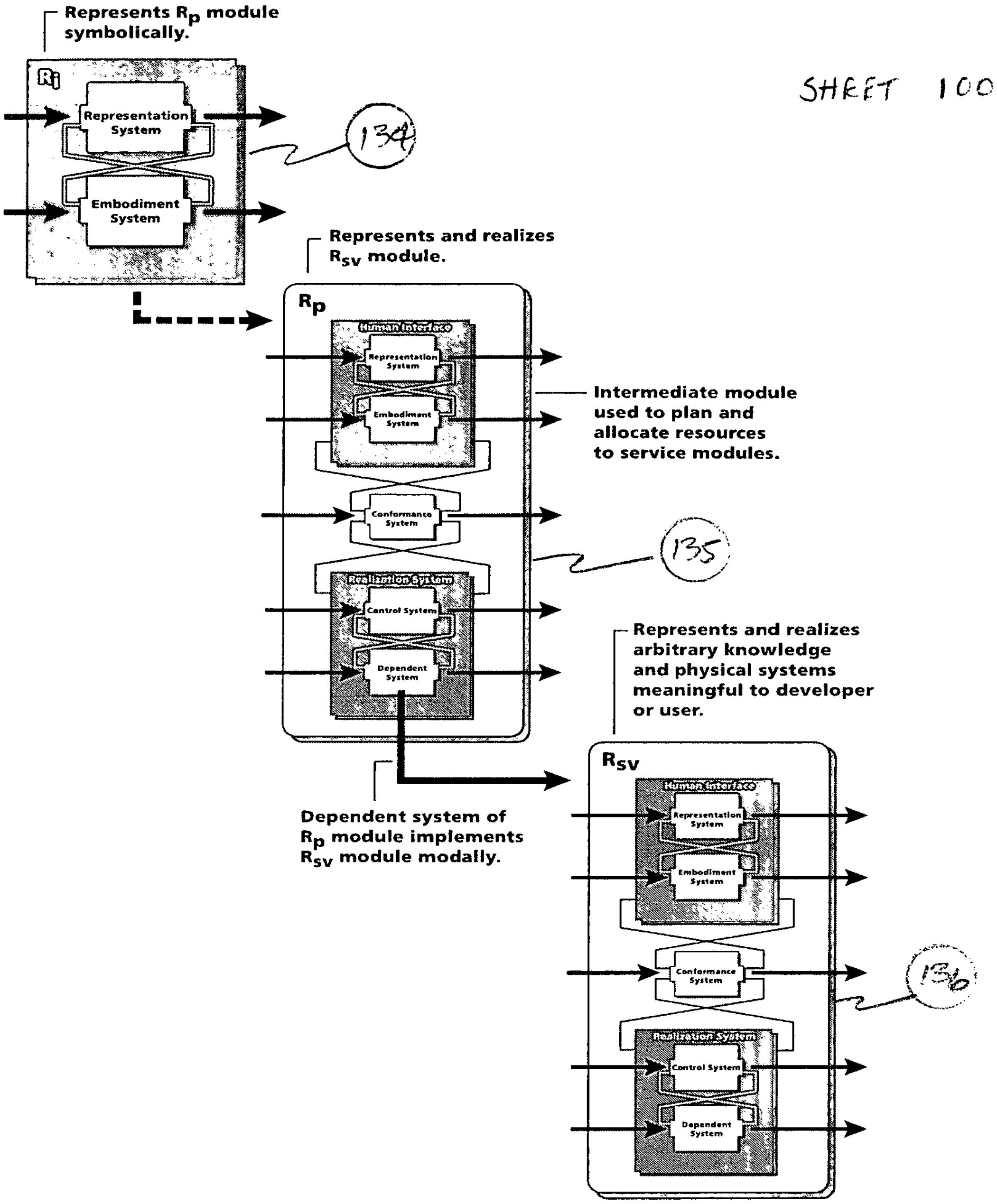

Intelligent "workstation" module of continuum configured as a resultant set-theoretic system of GSM. Defines five levels of continuum connectivity and is modeled according to a mind-body dualism
Collectively, the mind (embodiment system) and its ability to communicate through physical senses/ motors
Rg module
128
126
Human Interface
Sensory embodiment of physical symbols of language (e.g., the physical communicative senses/ motors)
Representation System
Levels of continuum connectivity
127
Cognitive embodiment of language (e.g., the mind)
Embodiment System
129
Maintains linguistic correspondences between cognitive (linguistic) model in HI and real physical form of RS
Conformance System
125
131
133
Realization System
Measures and controls real physical forms
Control System
130
Actual physical form realized by Rg module
Dependent System
132
Realizes and controls physical form according to cognitive model in HI

Fig. 95



Represents R_p module symbolically.

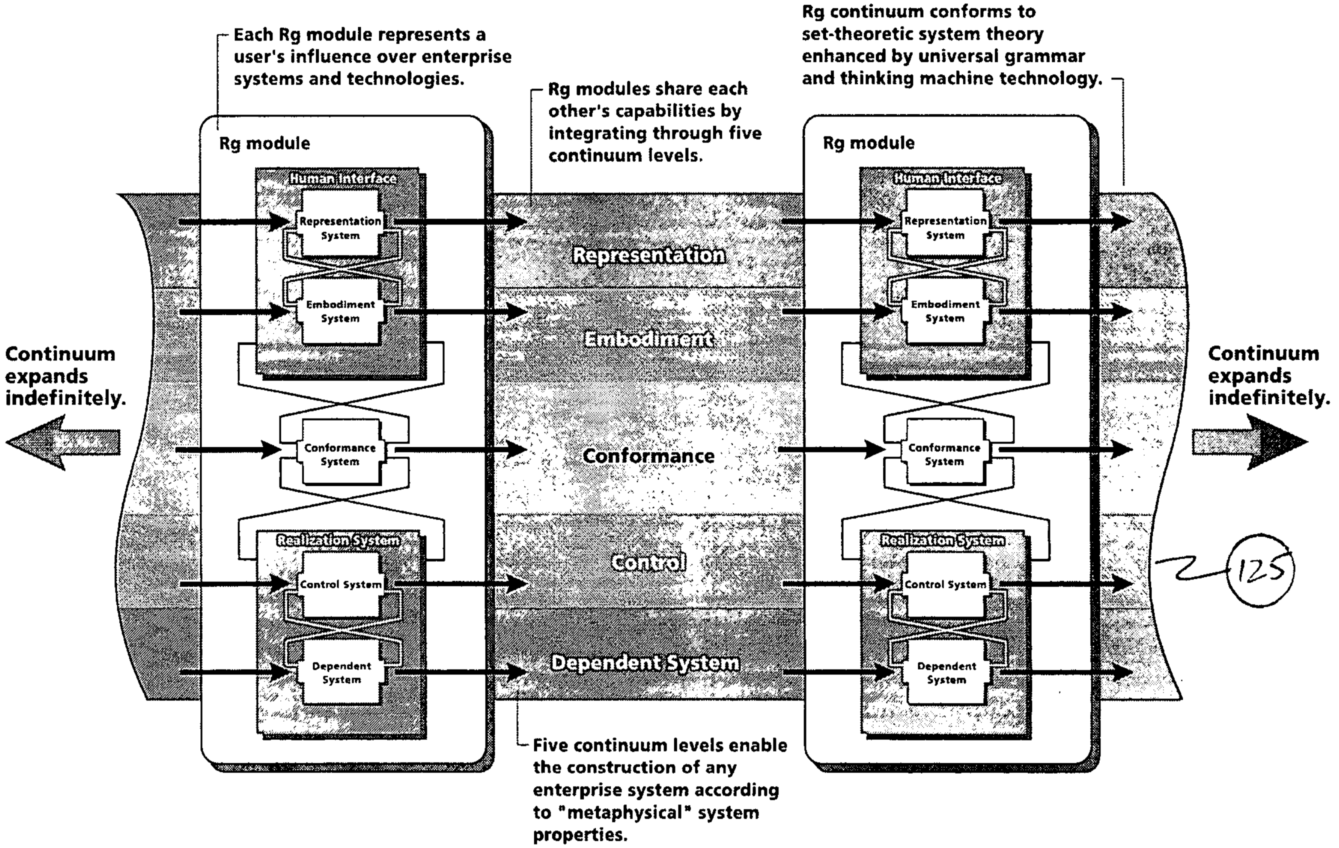
Represents and realizes R_{sv} module.

Intermediate module used to plan and allocate resources to service modules.

Represents and realizes arbitrary knowledge and physical systems meaningful to developer or user.

Dependent system of R_p module implements R_{sv} module modally.

Fig. 96



Each Rg module represents a user's influence over enterprise systems and technologies.

Rg modules share each other's capabilities by integrating through five continuum levels.

Rg continuum conforms to set-theoretic system theory enhanced by universal grammar and thinking machine technology.

Five continuum levels enable the construction of any enterprise system according to "metaphysical" system properties.

Fig. 97

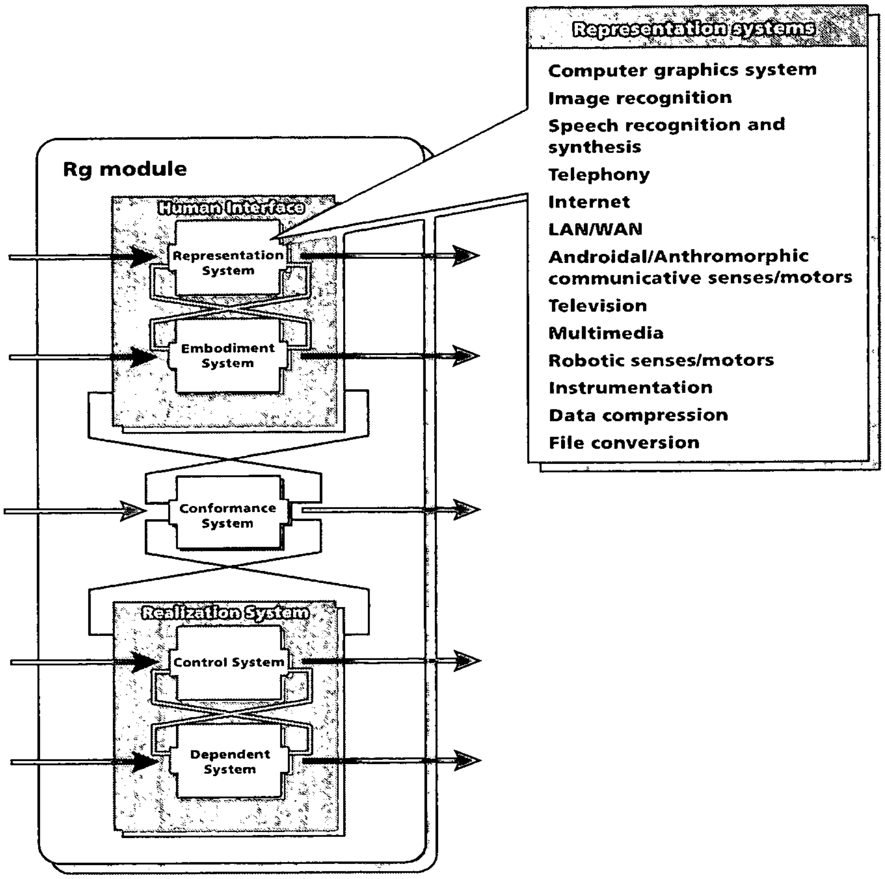

Rg module
Human Interface
Representation System
Embodiment System
Conformance System
Realization System
Control System
Dependent System
Representation systems
Computer graphics system
Image recognition
Speech recognition and synthesis
Telephony
Internet
LAN/WAN
Androidal/Anthromorphic communicative senses/motors
Television
Multimedia
Robotic senses/motors
Instrumentation
Data compression
File conversion

Fig. 98

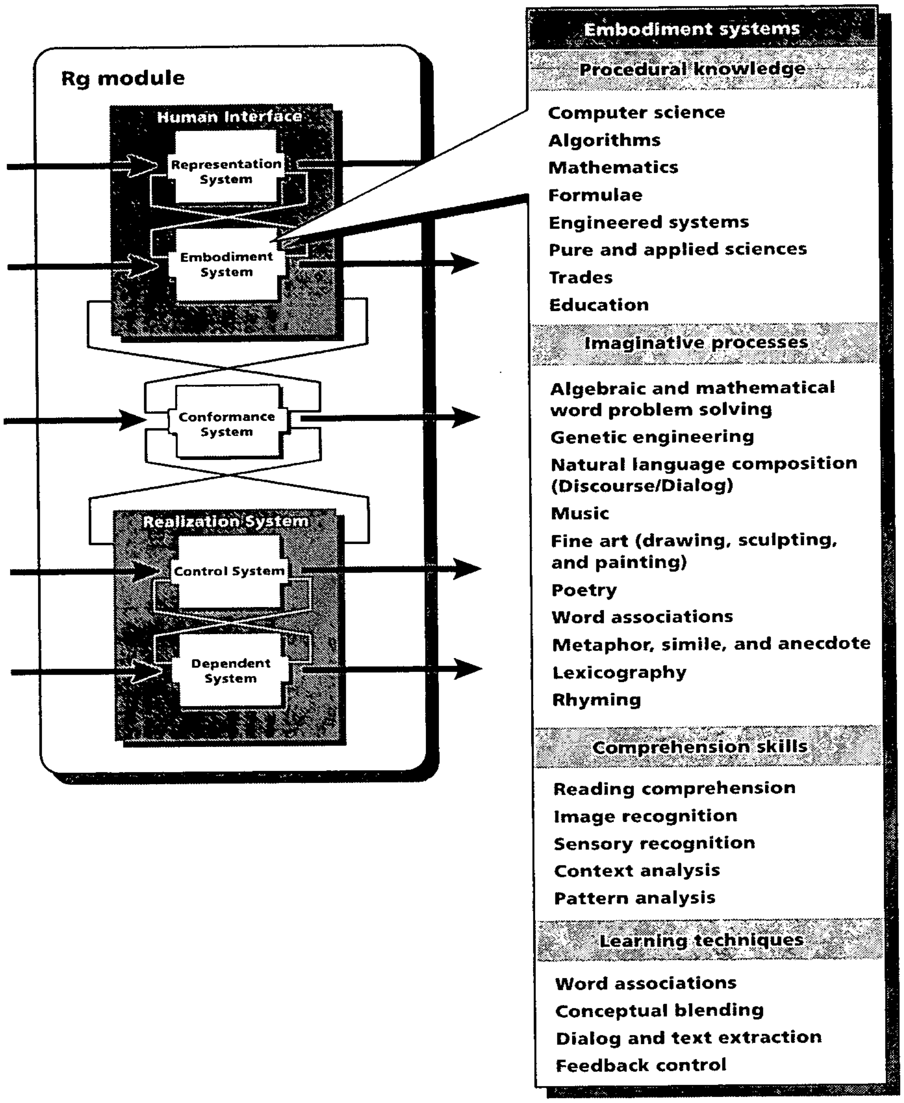

Rg module
Human Interface
Representation System
Embodiment System
Conformance System
Realization System
Control System
Dependent System
Embodiment systems
Procedural knowledge
Computer science
Algorithms
Mathematics
Formulae
Engineered systems
Pure and applied sciences
Trades
Education
Imaginative processes
Algebraic and mathematical word problem solving
Genetic engineering
Natural language composition (Discourse/Dialog)
Music
Fine art (drawing, sculpting, and painting)
Poetry
Word associations
Metaphor, simile, and anecdote
Lexicography
Rhyming
Comprehension skills
Reading comprehension
Image recognition
Sensory recognition
Context analysis
Pattern analysis
Learning techniques
Word associations
Conceptual blending
Dialog and text extraction
Feedback control

Fig. 99

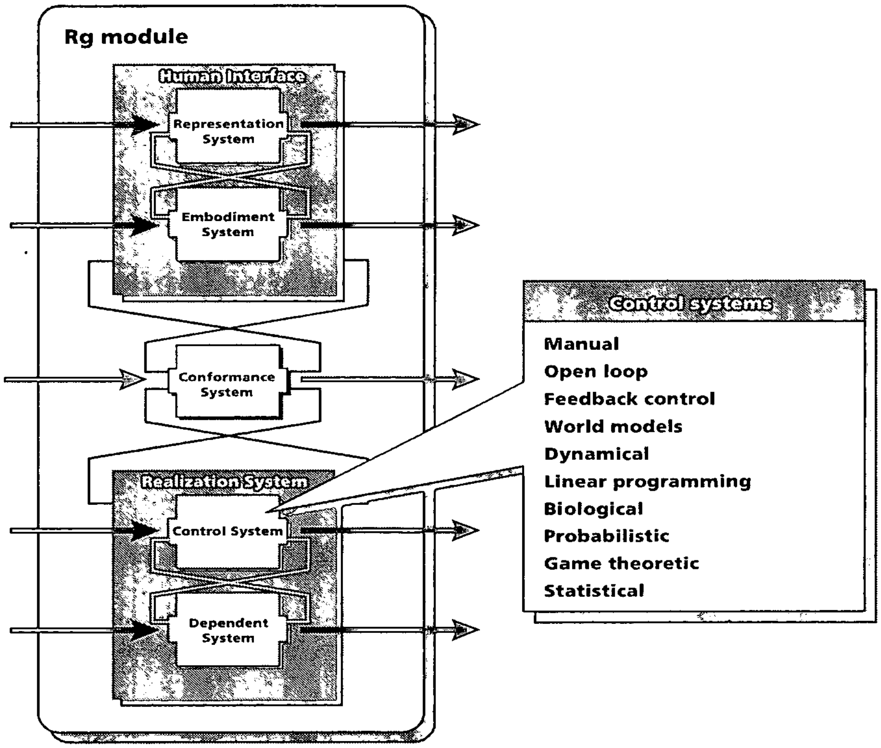

Rg module
Human Interface
Representation System
Embodiment System
Conformance System
Realization System
Control System
Dependent System
Control systems
Manual
Open loop
Feedback control
World models
Dynamical
Linear programming
Biological
Probabilistic
Game theoretic
Statistical

Fig. 100

SHEET 105 OF 140

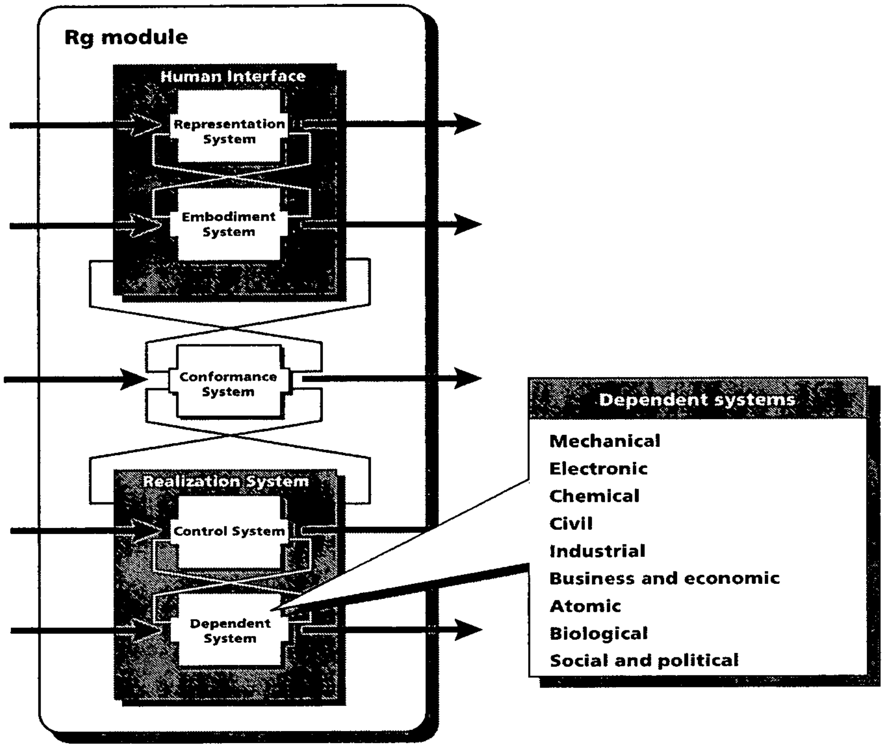


Fig. 101

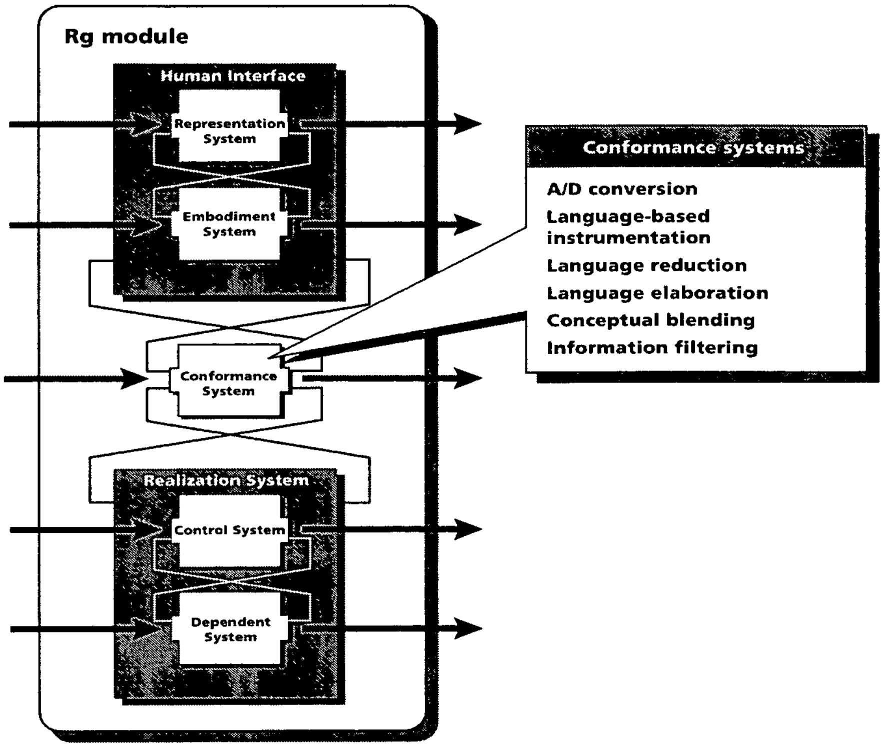

Rg module
Human Interface
Representation System
Embodiment System
Conformance System
Realization System
Control System
Dependent System
Conformance systems
A/D conversion
Language-based instrumentation
Language reduction
Language elaboration
Conceptual blending
Information filtering

Fig. 102

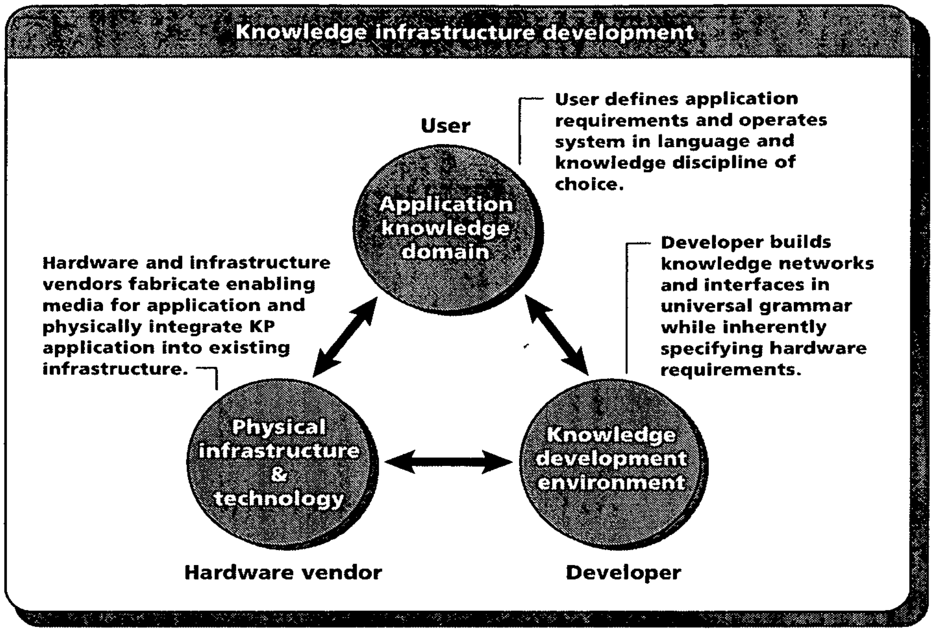
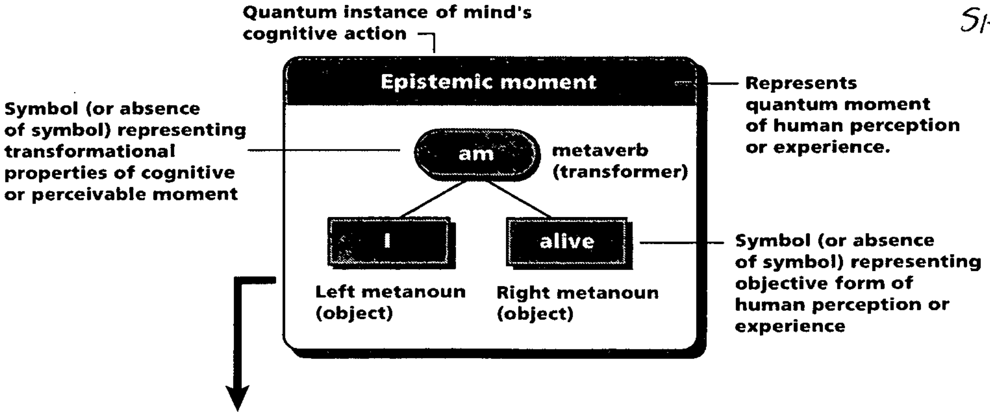

Knowledge infrastructure development
User
Application knowledge domain
User defines application requirements and operates system in language and knowledge discipline of choice.
Hardware and infrastructure vendors fabricate enabling media for application and physically integrate KP application into existing infrastructure.
Developer builds knowledge networks and interfaces in universal grammar while inherently specifying hardware requirements.
Physical infrastructure & technology
Knowledge development environment
Hardware vendor
Developer

Fig. 103

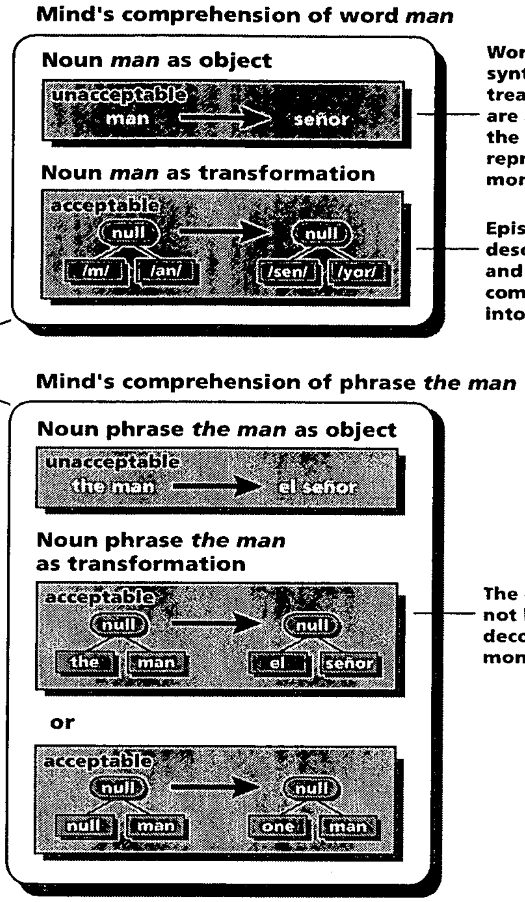

Quantum instance of mind's cognitive action
Epistemic moment
Represents quantum moment of human perception or experience.
Symbol (or absence of symbol) representing transformational properties of cognitive or perceivable moment
am
metaverb (transformer)
I
alive
Symbol (or absence of symbol) representing objective form of human perception or experience
Left metanoun (object)
Right metanoun (object)

Symbolic expressions representing epistemic moments

Grammatical property	Left metanoun	Metaverb	Right metanoun
Verb	I	am	alive
Adjective	brown	Blank space	cat
Composition	Sentence*	Period	Sentence*
Function	y	= f ( )	x
Inequality	A	>	B
Set	A	∈	B
Conjunction	a	AND	b
Alternative	a	OR	b
Negation	a	NOT	b
Matter	E	=	mc^2
Reaction	$2Hg^{2+}O^{2-*}$	$\xrightarrow{\Delta}$	$2Hg^0+O_2{}^{0*}$
Half-life	$e^{-\lambda t*}$	=	½ *
Dotted quarter note	♩	Null	·
Image	Shape, color, or texture A	Null	Shape, color, or texture B

* Transformations expressed as objective compositions are construed as single objects that are further deconstructed into respective epistemic moments.

Fig. 104


Mind's comprehension of word *man*
Noun *man* as object
unacceptable
man
señor
Noun *man* as transformation
acceptable
null
null
/m/
/an/
/sen/
/yor/
Mind's comprehension of phrase *the man*
Noun phrase *the man* as object
unacceptable
the man
el señor
Noun phrase *the man* as transformation
acceptable
null
null
the
man
el
señor
or
acceptable
null
null
null
man
one
man

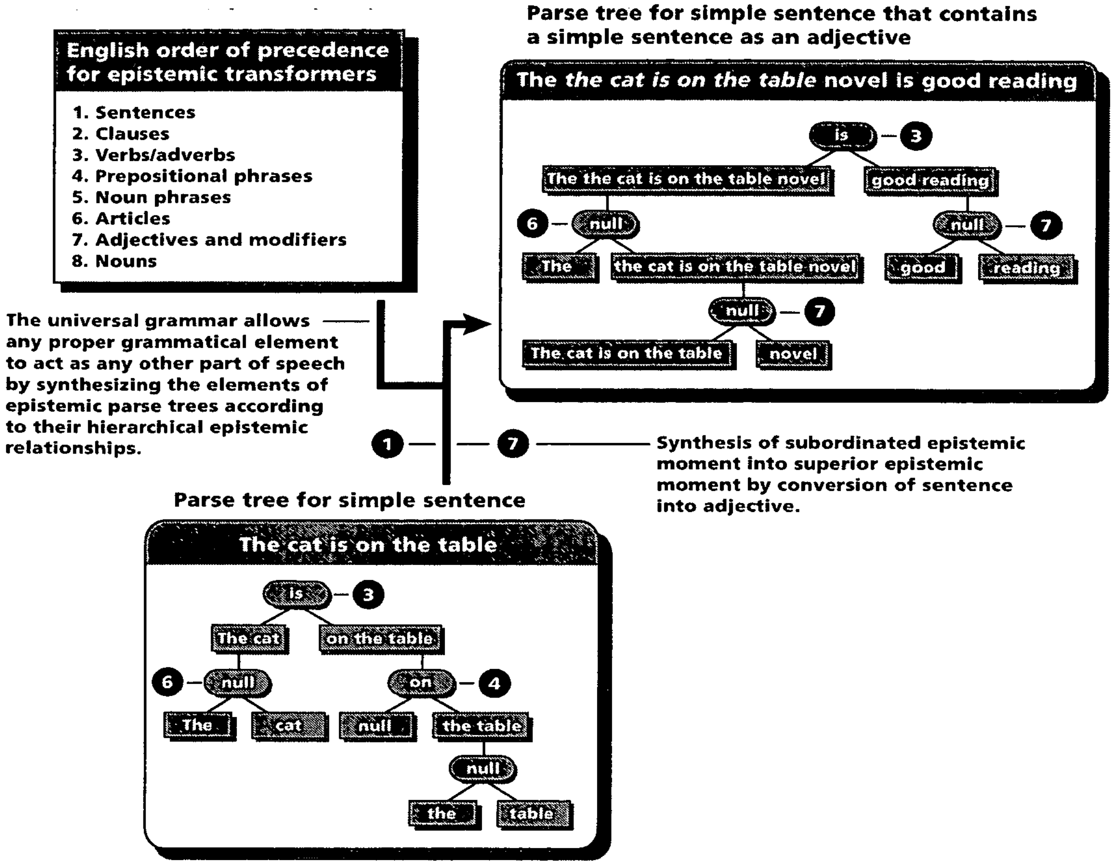
Words, phrases, and other syntactical constructions treated as linguistic objects are associations made *through* the mind's action and do not represent epistemic moments.

Epistemic transformations describe the mind's action and therefore can be comprehended and translated into other expressions.

The mind comprehends transformations of meaningful objects but not the meaning of the objects themselves.

The expression *the man* does not have meaning until it is deconstructed into an epistemic moment.

Fig. 105



English order of precedence for epistemic transformers

1. Sentences
2. Clauses
3. Verbs/adverbs
4. Prepositional phrases
5. Noun phrases
6. Articles
7. Adjectives and modifiers
8. Nouns

The universal grammar allows any proper grammatical element to act as any other part of speech by synthesizing the elements of epistemic parse trees according to their hierarchical epistemic relationships.

Parse tree for simple sentence that contains a simple sentence as an adjective

Synthesis of subordinated epistemic moment into superior epistemic moment by conversion of sentence into adjective.

Parse tree for simple sentence

Fig. 106

Epistemic moment is used as the basis of a metaphoric translation by exchanging the right metanouns of the subject and object of a simile.

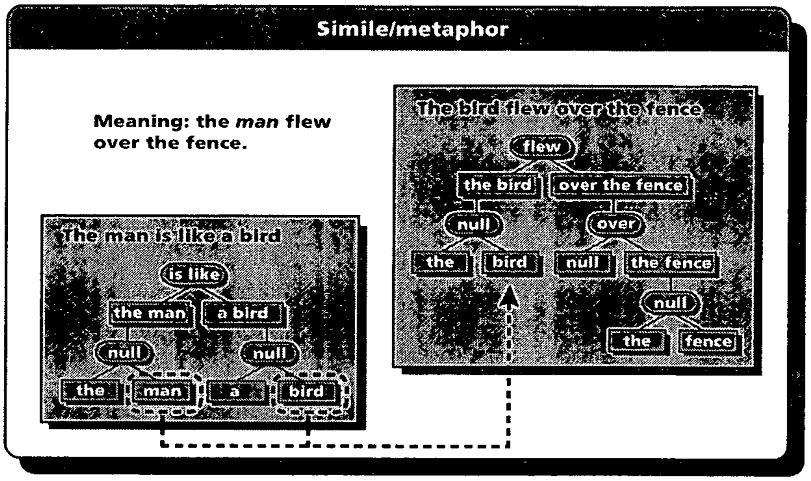


Fig. 107

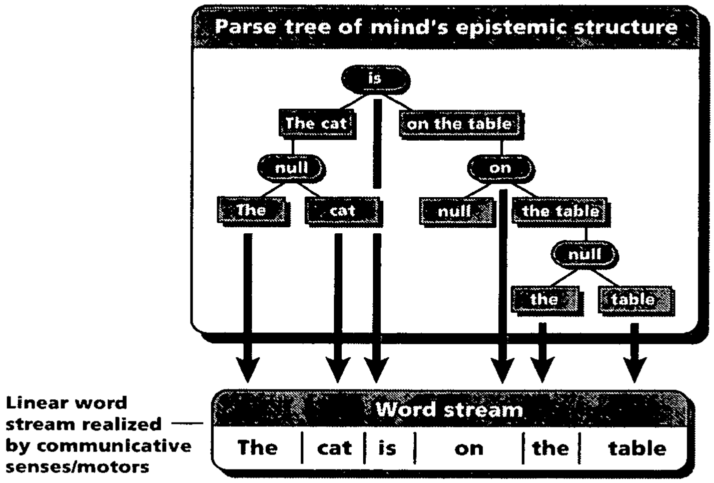
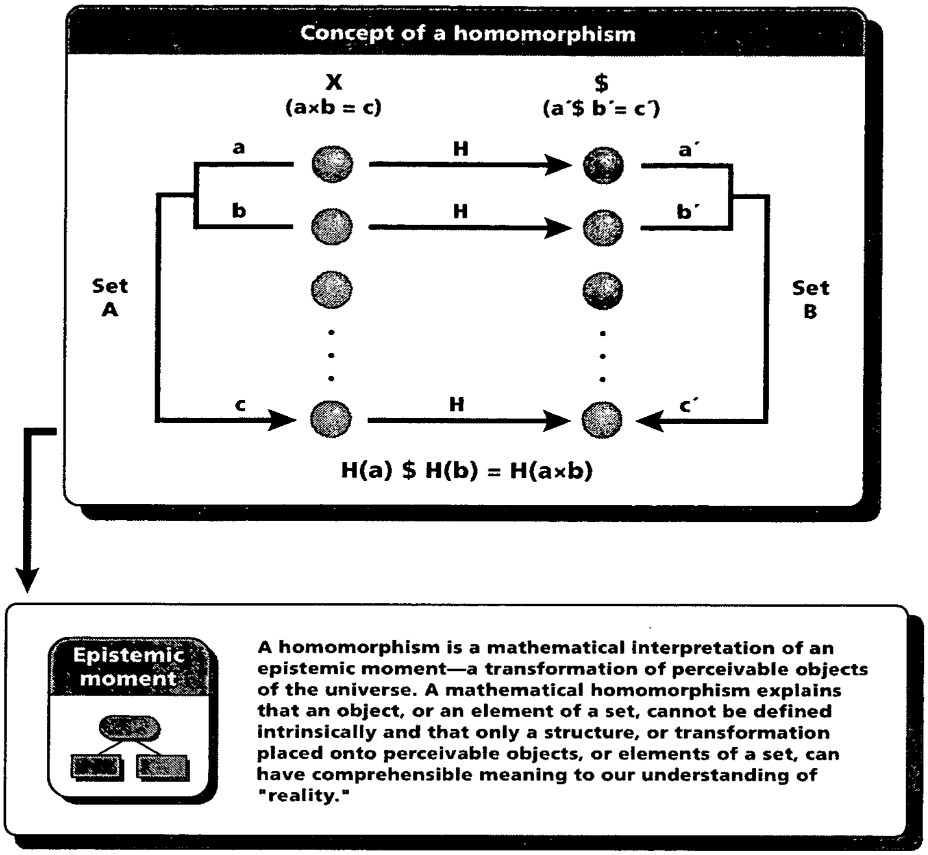

Parse tree of mind's epistemic structure
is
The cat
on the table
null
on
The
cat
null
the table
null
the
table
Linear word stream realized by communicative senses/motors
Word stream
The
cat
is
on
the
table

Fig. 108



A homomorphism is a mathematical interpretation of an epistemic moment—a transformation of perceivable objects of the universe. A mathematical homomorphism explains that an object, or an element of a set, cannot be defined intrinsically and that only a structure, or transformation placed onto perceivable objects, or elements of a set, can have comprehensible meaning to our understanding of "reality."

Fig. 109

SHEET 114 OF 140

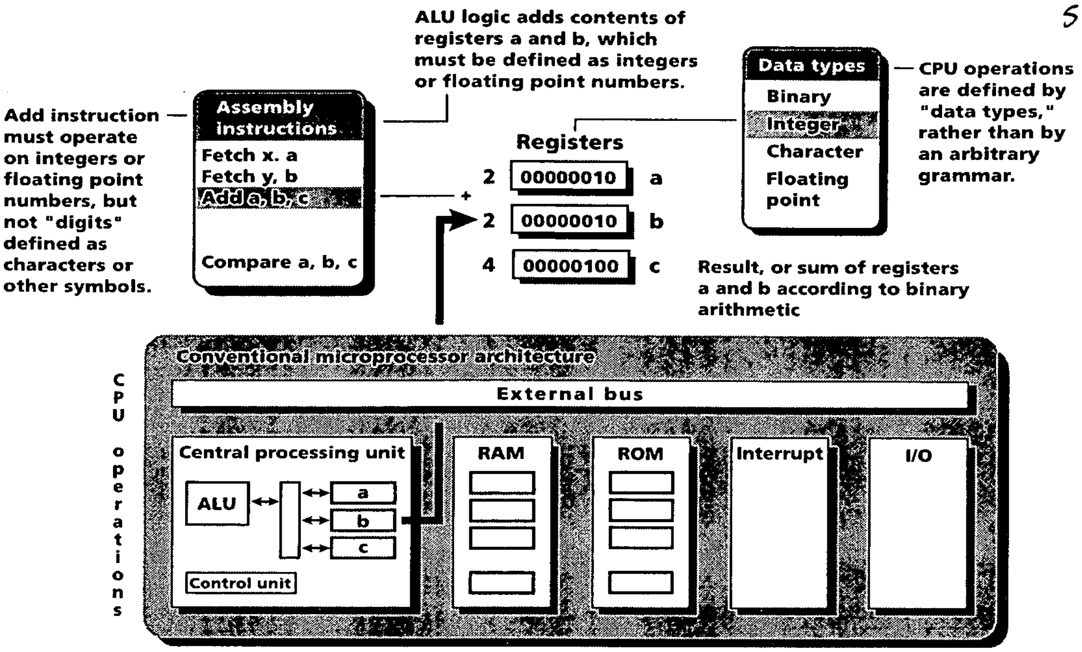
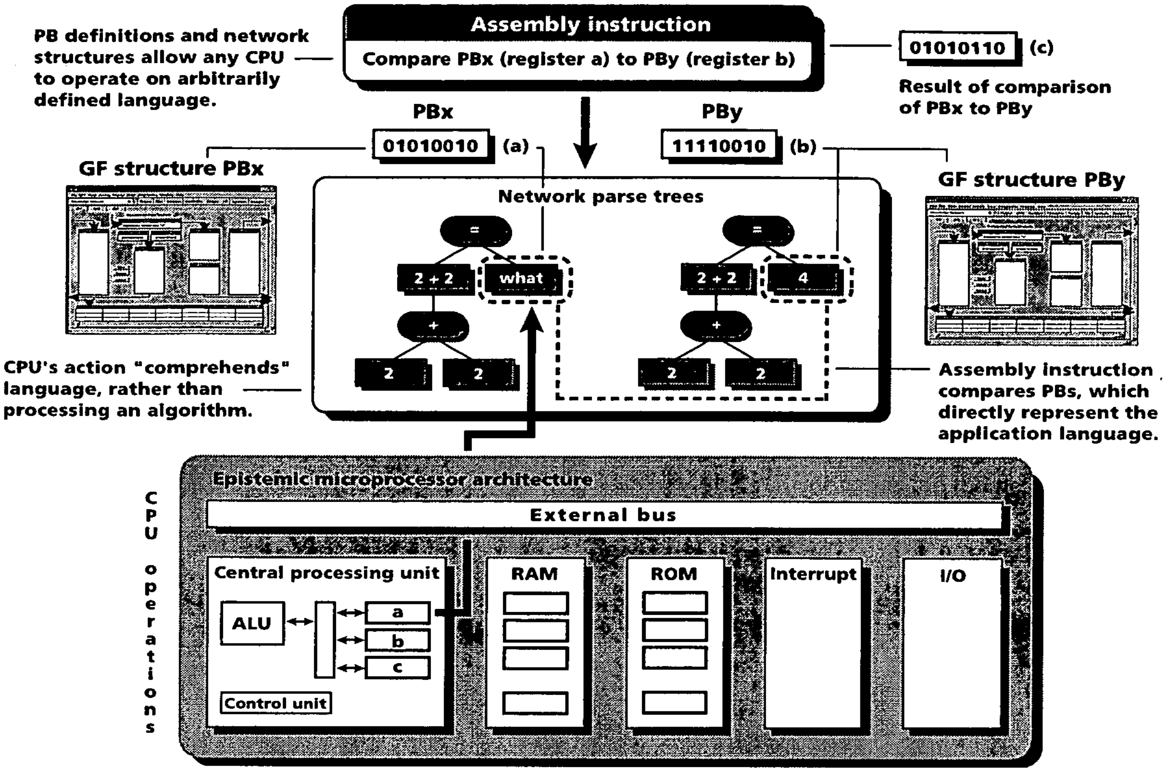

ALU logic adds contents of registers a and b, which must be defined as integers or floating point numbers.
Add instruction must operate on integers or floating point numbers, but not "digits" defined as characters or other symbols.
Assembly instructions
Fetch x. a
Fetch y, b
Add a, b, c
Compare a, b, c
Registers
2
00000010
a
+
2
00000010
b
4
00000100
c
Data types
Binary
Integer
Character
Floating point
CPU operations are defined by "data types," rather than by an arbitrary grammar.
Result, or sum of registers a and b according to binary arithmetic
CPU operations
Conventional microprocessor architecture
External bus
Central processing unit
ALU
a
b
c
Control unit
RAM
ROM
Interrupt
I/O

Fig. 110



PB definitions and network structures allow any CPU to operate on arbitrarily defined language.

Result of comparison of PBx to PBy

CPU's action "comprehends" language, rather than processing an algorithm.

Assembly instruction compares PBs, which directly represent the application language.

Fig. 111

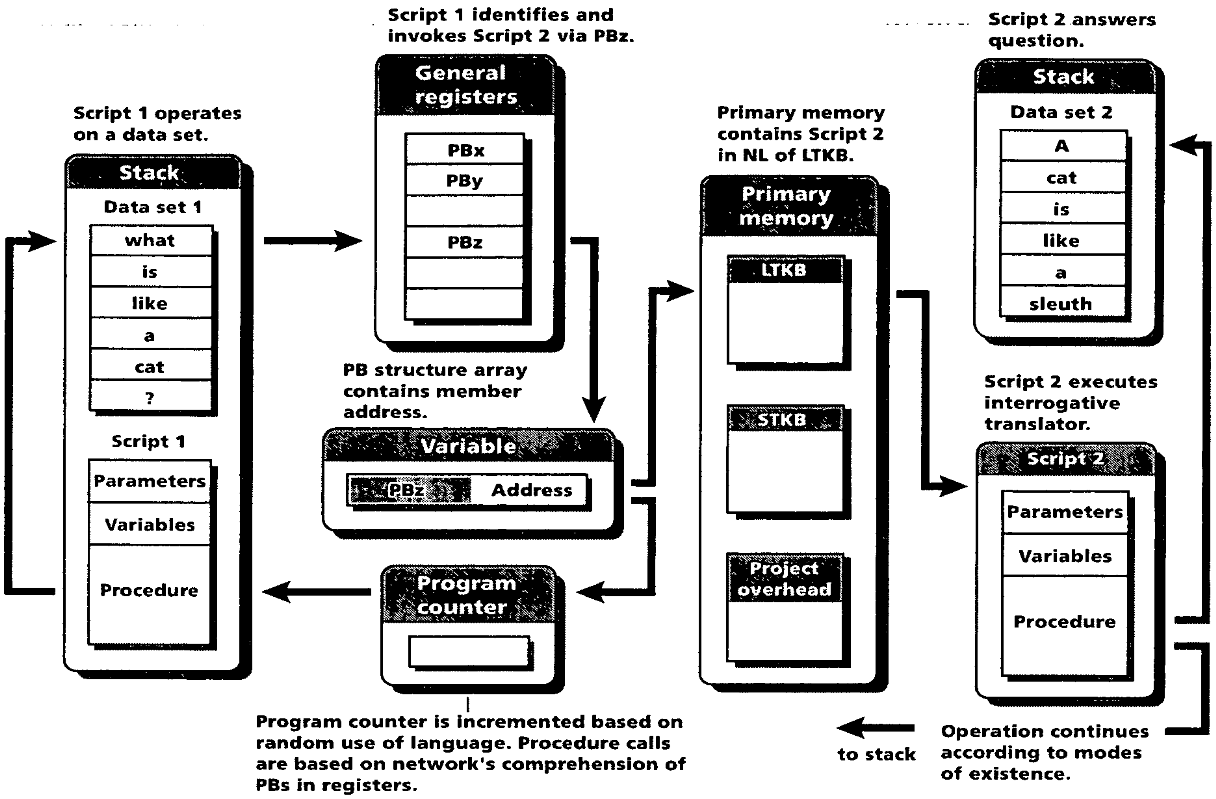
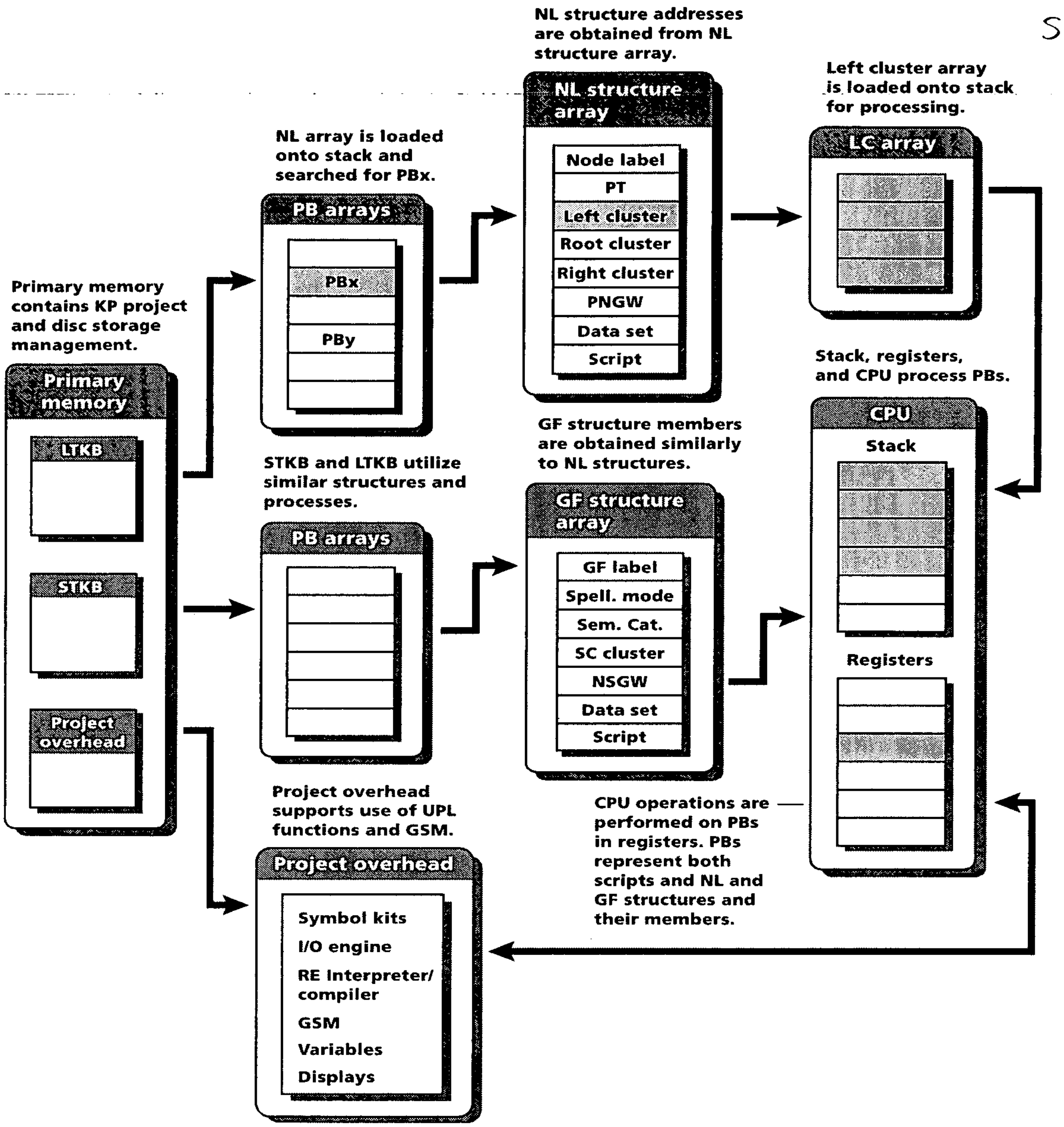

Script 1 operates on a data set.
Stack
Data set 1
what
is
like
a
cat
?
Script 1
Parameters
Variables
Procedure
Script 1 identifies and invokes Script 2 via PBz.
General registers
PBx
PBy
PBz
PB structure array contains member address.
Variable
PBz
Address
Program counter
Program counter is incremented based on random use of language. Procedure calls are based on network's comprehension of PBs in registers.
Primary memory contains Script 2 in NL of LTKB.
Primary memory
LTKB
STKB
Project overhead
to stack
Script 2 answers question.
Stack
Data set 2
A
cat
is
like
a
sleuth
Script 2 executes interrogative translator.
Script 2
Parameters
Variables
Procedure
Operation continues according to modes of existence.

Fig. 112

SHEET 117 OF 140



NL structure addresses are obtained from NL structure array.
Left cluster array is loaded onto stack for processing.
NL array is loaded onto stack and searched for PBx.
NL structure array
Node label
PT
Left cluster
Root cluster
Right cluster
PNGW
Data set
Script
LC array
PB arrays
PBx
PBy
Primary memory contains KP project and disc storage management.
Primary memory
LTKB
STKB
Project overhead
Stack, registers, and CPU process PBs.
CPU
Stack
Registers
GF structure members are obtained similarly to NL structures.
STKB and LTKB utilize similar structures and processes.
GF structure array
GF label
Spell. mode
Sem. Cat.
SC cluster
NSGW
Data set
Script
PB arrays
Project overhead supports use of UPL functions and GSM.
CPU operations are performed on PBs in registers. PBs represent both scripts and NL and GF structures and their members.
Project overhead
Symbol kits
I/O engine
RE Interpreter/ compiler
GSM
Variables
Displays

Fig. 113

SHEET 118 OF 140

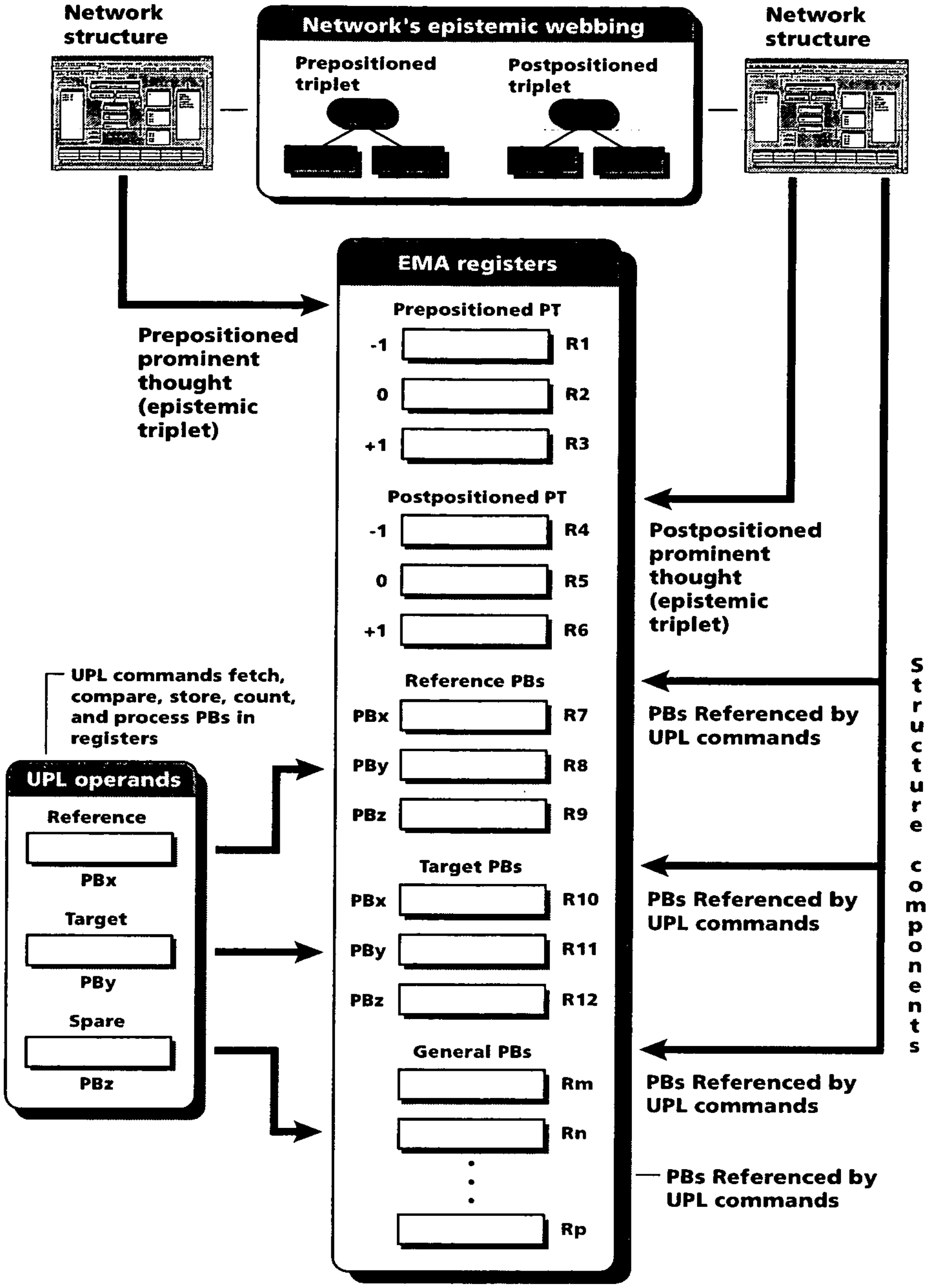

Network structure
Network's epistemic webbing
Prepositioned triplet
Postpositioned triplet
Network structure
EMA registers
Prepositioned PT
-1
R1
0
R2
+1
R3
Prepositioned prominent thought (epistemic triplet)
Postpositioned PT
-1
R4
0
R5
+1
R6
Postpositioned prominent thought (epistemic triplet)
UPL commands fetch, compare, store, count, and process PBs in registers
Reference PBs
PBx
R7
PBy
R8
PBz
R9
PBs Referenced by UPL commands
Structure components
UPL operands
Reference
PBx
Target
PBy
Spare
PBz
Target PBs
PBx
R10
PBy
R11
PBz
R12
PBs Referenced by UPL commands
General PBs
Rm
Rn
Rp
PBs Referenced by UPL commands
PBs Referenced by UPL commands

Fig. 114

ASCII/Unicode PB bit fields for symbol kits

NL designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var	Sub-gram Form y Var	Root word	Root word Var	Display Protocol	Root word ID
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GF designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var	Sub-gram Form y Var	Root word	Root word Var	Display Protocol	Root word ID
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

ASCII/Unicode NL

1. NL designator used to Read/Write root node of ASCII byte/Unicode multibyte to and from external machines. Designates ASCII/Unicode byte structure before it is interpreted as a GF structure of an application language.

2. Designates external hardware or software protocol using ASCII/Unicode byte structure. Also specifies GSM system element.

3. Designates intellectual faculties using ASCII node structure.

4. ASCII/Unicode byte structure interpreted as machine language element transformer. Can be used to integrate ASCII byte with other machine languages, such as executable code.

5. Use level 3, leaving level 0 for external sensory structures, level 1 for bits (0 and 1), level 2 for bit fields, and level 4 for application lexicography.

6. Specifies root-node transformation of leftmost bit with 7 rightmost bits. Other bits transform in parse-tree hierarchy corresponding to bit field nodal transformations (i.e., four rightmost bits with remaining leftmost bits, etc.).

7. Not applicable to most text files, except GF variant may be used to designate EBIDIC and other text file types if leftmost parity is employed.

8. Not applicable, but can be used for situations in which leftmost bit transforms with remaining rightmost bits for reasons other than bit parity.

9. Not applicable, but can be used for situations in which leftmost bit transforms with remaining rightmost bits for reasons other than bit parity.

10. Not applicable, but can be used for situations in which leftmost bit transforms with remaining rightmost bits for reasons other than bit parity.

11. Not applicable, but can be used for situations in which leftmost bit transforms with remaining rightmost bits for reasons other than bit parity.

12. Designates topical semantic category of root node transformation of ASCII byte or Unicode multibyte.

13. Not applicable, but can be used when multiple interpretations of root node are necessary.

14. Designates protocols that display root node transformation, usually in connection with compilers and linkers.

15. Identifies NL structure according to configuration control number, primary key encoding, or simple numerical sequence.

ASCII/Unicode GF

16. Each GF designator defines an alternative use of the ASCII/Unicode root-node transformer. Possible uses include ASCII/Unicode byte; natural language alphanumeric character (a, b, c, d, . . . 1, 2, 3, 4, etc.); musical note; EDI character; pixel image element; or any other linguistic element embedded in the byte structure by hardware or software vendor.

17. Designates external hardware or software protocol using ASCII/Unicode byte structure. Also specifies GSM system element.

18. Designates intellectual faculties using ASCII node structure.

19. Designates language of embedded element when implemented in ASCII/Unicode text.

20. Use level 4 to begin embedded language lexicography (i.e., for character "a," number "1," etc.).

21. Designates grammatical form of embedded language element, including "character," "number," etc. (Also can be used to designate character's location in syntax, such as 1st character a in word, etc.)

22. Designates variant of embedded character, such as typeface.

23. Designates alternative syntactical uses of embedded character, such as vowel sounds and digraphs ("ch").

24. Designates alternative syntactical uses of embedded character, such as vowel sounds and digraphs ("ch").

25. Designates alternative syntactical uses of embedded character, such as vowel sounds and digraphs ("ch").

26. Designates alternative syntactical uses of embedded character, such as vowel sounds and digraphs ("ch").

27. Semantically classifies character, number, or other symbol used in ASCII/Unicode standard.

28. Designates semantic category variant.

29. External and Host machine displays used for particular character and its variants.

30. Identifies GF structure according to configuration control number, primary key encoding, or simple numerical sequence.

Fig. 115

Macrocode PB bit fields for symbol kits

NL designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var	Sub-gram Form y Var	Root word	Root word Var	Display Protocol	Root word ID
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GF designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var	Sub-gram Form y Var	Root word	Root word Var	Display Protocol	Root word ID
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Macrocode NL

1. Designates root-node transformation of executable byte used on external hardware. Reading or Writing the NL allows the knowledge network to process the external byte as a node structure before it obtains higher-level definition in the machine language as a GF structure.

2. Designates external hardware or software protocol using macrocode byte structure, or configures byte structure as GSM system element.

3. Designates knowledge disciplines pertinent to machine code processing, such as processor design and architecture, compiler design, and Boolean algebra.

4. Defines architecture type and design methodologies. Describes elements of digital circuits and microprocessor logic as language elements.

5. Use level 1 for bits, level 2 for bit fields, level 3 for bytes, and level 4 for byte structures and embedded languages.

6. Designates root-node transformation of executable byte, such as the synthesis of an instruction's bit sequence with the enabling control signals of the byte.

7. Designates grammatical properties of root node transformer.

8. Not applicable, but can be used for situations in which root node transformer may be classified by alternative grammatical interpretations.

9. Not applicable, but can be used for situations in which root node transformer may be classified by alternative grammatical interpretations.

10. Not applicable, but can be used for situations in which root node transformer may be classified by alternative grammatical interpretations.

11. Not applicable, but can be used for situations in which root node transformer may be classified by alternative grammatical interpretations.

12. Designates semantic category of root node transformer of executable byte. Examples include indirect and implied memory addresses, instruction or data bit fields, and specialized data structures such as pointers and variables.

13. Not applicable, but can be used when multiple interpretations of root node category are necessary.

14. Designates protocols that display root node transformation, usually in connection with compilers and linkers.

15. Identifies NL structure according to configuration control number, primary key encoding, or simple numerical sequence.

Macrocode GF

16. Each GF designator defines an alternative use of the macrocode instruction or data. Possible uses include primary memory's "load register a" instructions, and direct and implied memory addressing.

17. Designates external hardware or software protocol using ASCII/Unicode byte structure, or configures byte structure as GSM system element.

18. Designates intellectual faculties using macrocode GF structure.

19. Designates language used to specify microprocessor or digital logic operations or data.

20. Use level 4 for byte structures and embedded languages.

21. Designates grammatical form of embedded machine language, including memory fetch and store, I/O, interrupt, integer and floating point data, and stack operations.

22. Designates variant of embedded element, such as fetch a, b → load into register location a_1, or a_2, or a_n (the variant registers).

23. Designates sub-grammatical uses of instruction or data, such as those indicating memory device to be used.

24. Designates sub-grammatical uses of instruction or data, such as those indicating memory device to be used.

25. Designates sub-grammatical uses of instruction or data, such as those indicating memory device to be used.

26. Designates sub-grammatical uses of instruction or data, such as those indicating memory device to be used.

27. Semantically classifies macrocode instruction or data, such as "I/O instruction."

28. Used for semantic category variant.

29. Designates display of bit sequence or embedded language.

30. Identifies GF structure according to configuration control number, primary key encoding, or simple numerical sequence.

Fig. 116

Generalized PB bit fields for symbol kits

NL designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var.	Sub-gram Form y Var.	Root word	Root word Var	Display Protocol	Root word ID
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GF designator

PB Class	I/O System Vector	Knowledge Discipline	Language	Syntactical Level	Gram. Form	Gram. Form Var	Sub-gram Form x	Sub-gram Form y	Sub-gram Form x Var	Sub-gram Form y Var	Root word	Root word Var	Display Protocol	Root word ID
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Generalized NL/GF

1. Designates transformational structure of any external data when that data is analyzed as an epistemic parse tree.

16. Designates GF structure, or objective form of any external data. This data is usually interpreted as embedded language element.

2, 17. Designates any external protocol associated with NL structure, including system vector.

3, 18. Designates knowledge network's intellectual faculties (UPL functions) that normally process the given NL or GF.

4, 19. Designates the language in which the external structure is defined.

5, 20. Designates the syntactical level of the external structure once converted into knowledge network's PB structure.

6, 21. Designates any grammatical form of any language element.

7, 22. Designates any gf variant.

8, 23. Designates any matrix-related grammatical elaboration.

9, 24. Designates any matrix-related grammatical elaboration.

10, 25. Designates any matrix-related grammatical elaboration.

11, 26. Designates any matrix-related grammatical elaboration.

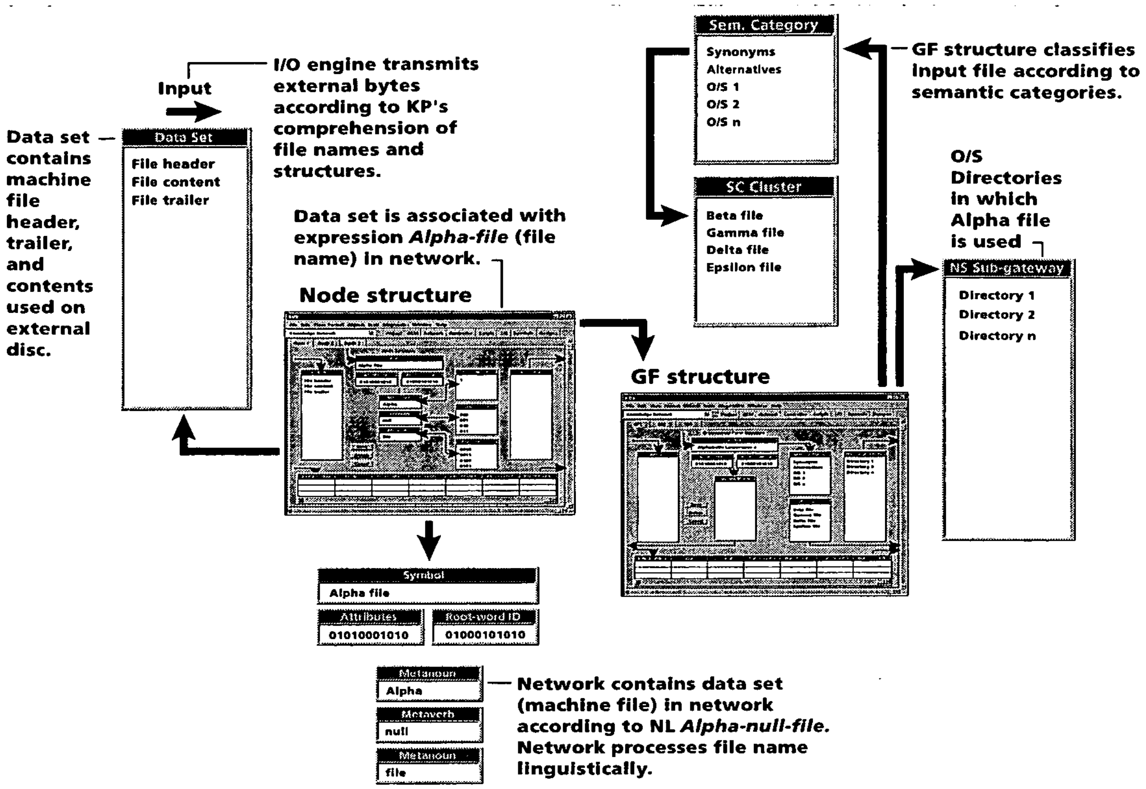
12, 27. Designates semantic category, or "topic" of external data structure.

13, 28. Designates semantic category variant.

14, 29. Designates Host or external system protocol that displays related symbol.

15, 30. Identifies NL or GF structure according to configuration control number, primary key, numerical sequence, or any other system of encoding used for language elements.

Fig. 117



I/O engine transmits external bytes according to KP's comprehension of file names and structures.

Data set contains machine file header, trailer, and contents used on external disc.

Data set is associated with expression *Alpha-file* (file name) in network.

GF structure classifies input file according to semantic categories.

O/S Directories in which Alpha file is used

Network contains data set (machine file) in network according to NL *Alpha-null-file.* Network processes file name linguistically.

Fig. 118

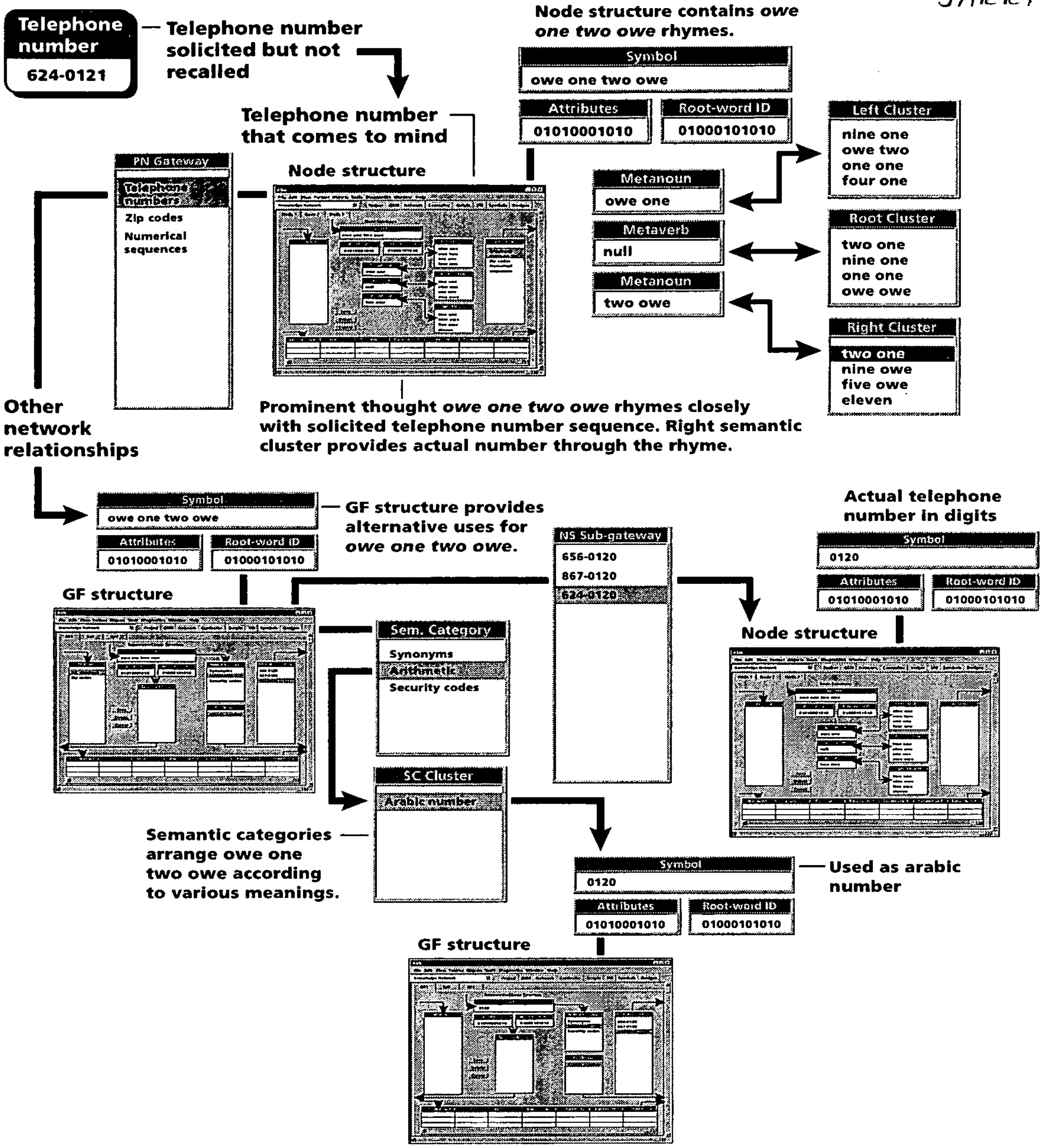

Telephone number
624-0121
— Telephone number solicited but not recalled
Telephone number that comes to mind
Node structure contains *owe one two owe* rhymes.
Symbol
owe one two owe
Attributes
01010001010
Root-word ID
01000101010
PN Gateway
Telephone numbers
Zip codes
Numerical sequences
Node structure
Metanoun
owe one
Metaverb
null
Metanoun
two owe
Left Cluster
nine one
owe two
one one
four one
Root Cluster
two one
nine one
one one
owe owe
Right Cluster
two one
nine owe
five owe
eleven
Other network relationships
Prominent thought *owe one two owe* rhymes closely with solicited telephone number sequence. Right semantic cluster provides actual number through the rhyme.
Symbol
owe one two owe
Attributes
01010001010
Root-word ID
01000101010
— GF structure provides alternative uses for *owe one two owe*.
GF structure
NS Sub-gateway
656-0120
867-0120
624-0120
Actual telephone number in digits
Symbol
0120
Attributes
01010001010
Root-word ID
01000101010
Node structure
Sem. Category
Synonyms
Arithmetic
Security codes
SC Cluster
Arabic number
Semantic categories arrange owe one two owe according to various meanings.
Symbol
0120
Attributes
01010001010
Root-word ID
01000101010
— Used as arabic number
GF structure

Fig. 119

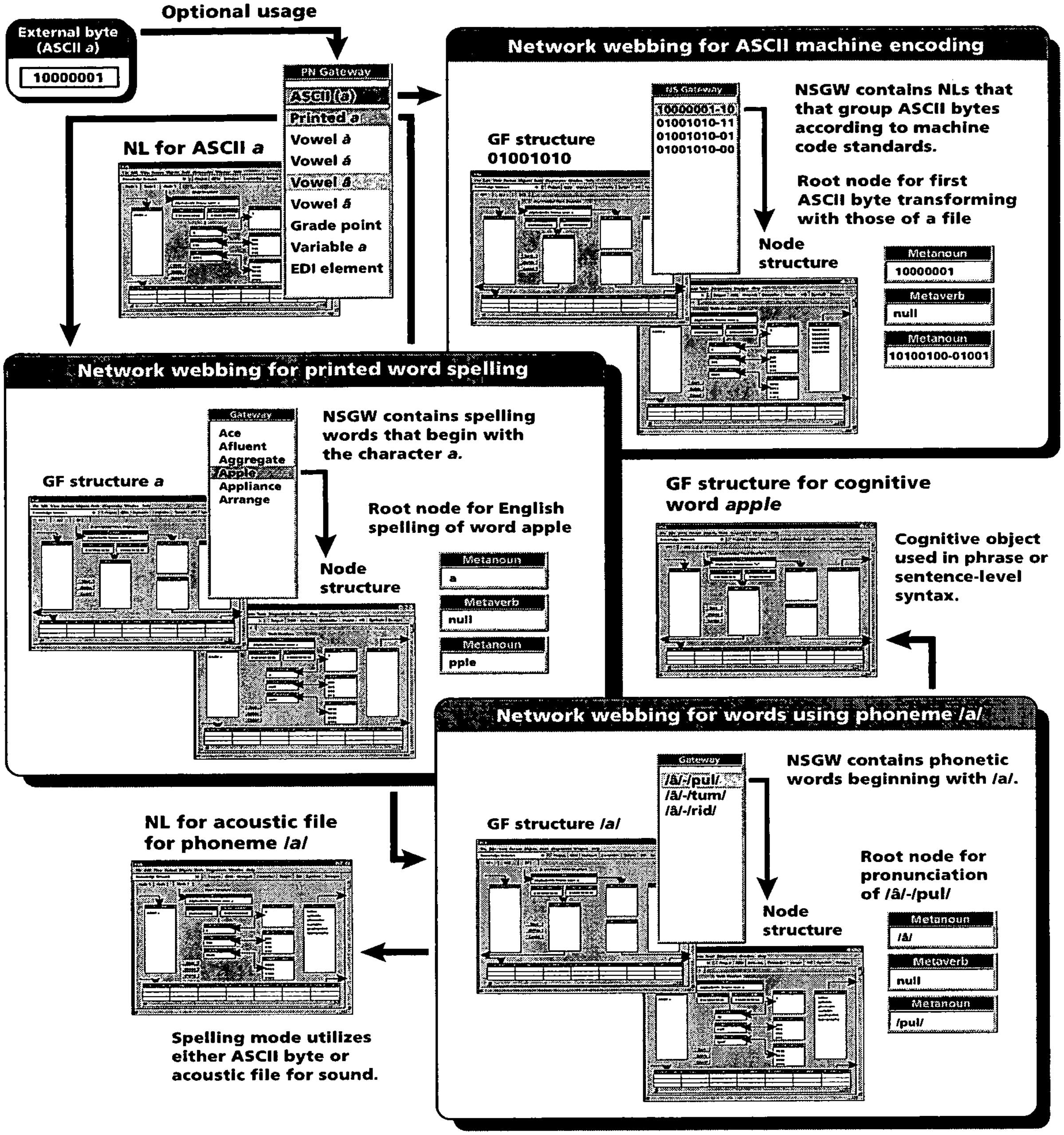

Optional usage
External byte (ASCII a)
10000001
PN Gateway
ASCII (a)
Printed a
Vowel à
Vowel á
Vowel â
Vowel ă
Grade point
Variable a
EDI element
NL for ASCII a
Network webbing for ASCII machine encoding
GF structure 01001010
NS Gateway
10000001-10
01001010-11
01001010-01
01001010-00
NSGW contains NLs that that group ASCII bytes according to machine code standards.
Root node for first ASCII byte transforming with those of a file
Node structure
Metanoun
10000001
Metaverb
null
Metanoun
10100100-01001
Network webbing for printed word spelling
Gateway
Ace
Afluent
Aggregate
Apple
Appliance
Arrange
NSGW contains spelling words that begin with the character a.
GF structure a
Root node for English spelling of word apple
Node structure
Metanoun
a
Metaverb
null
Metanoun
pple
GF structure for cognitive word apple
Cognitive object used in phrase or sentence-level syntax.
Network webbing for words using phoneme /a/
Gateway
/ă/-/pul/
/ă/-/tum/
/ă/-/rid/
NSGW contains phonetic words beginning with /a/.
GF structure /a/
Root node for pronunciation of /ă/-/pul/
Node structure
Metanoun
/ă/
Metaverb
null
Metanoun
/pul/
NL for acoustic file for phoneme /a/
Spelling mode utilizes either ASCII byte or acoustic file for sound.

Fig. 120

SHEET 125 OF 140

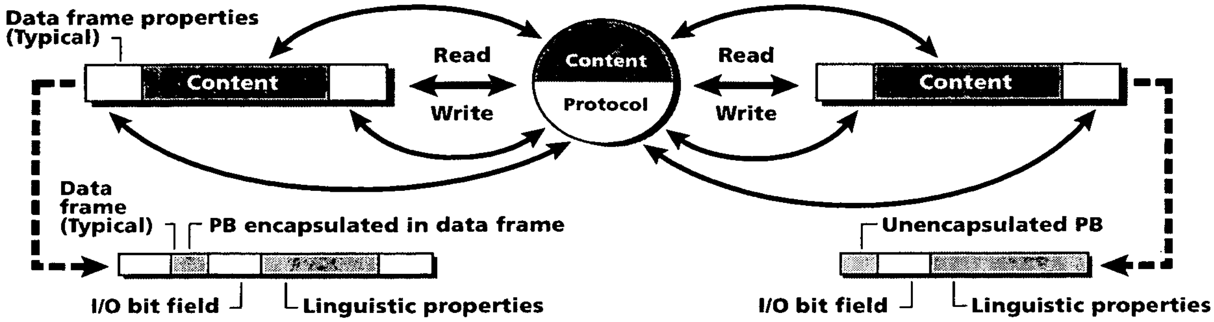


Fig. 121

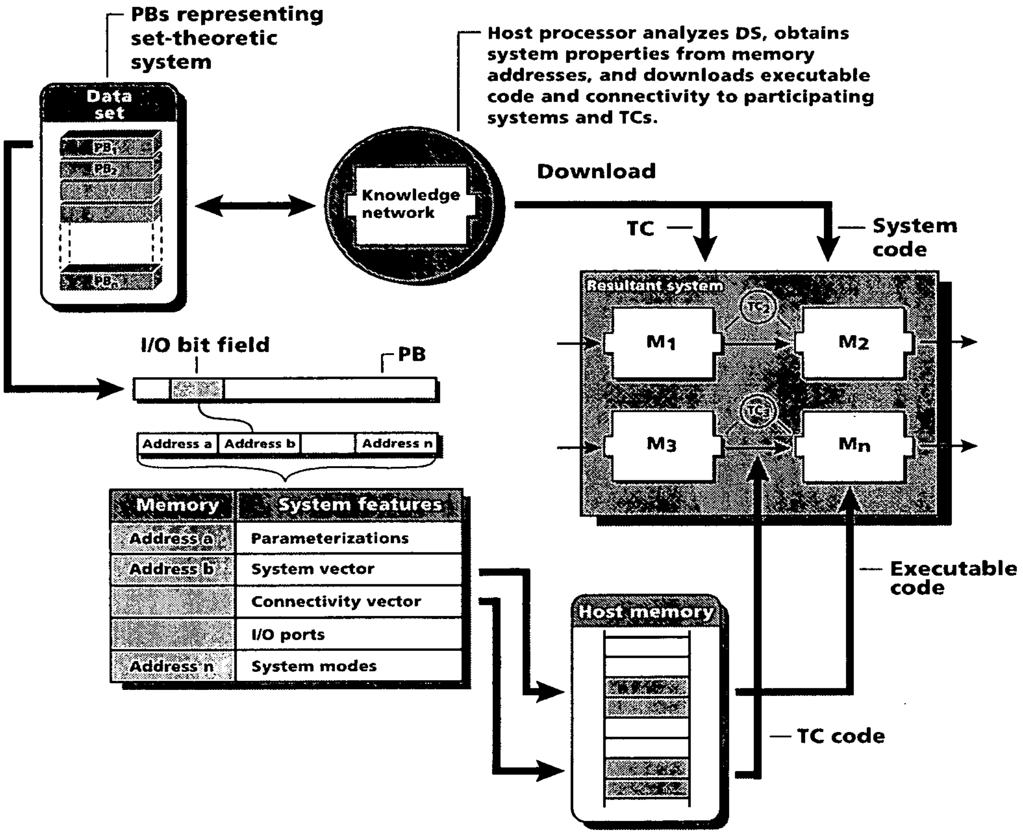

PBs representing set-theoretic system
Data set
PB1
PB2
PBn
Knowledge network
Host processor analyzes DS, obtains system properties from memory addresses, and downloads executable code and connectivity to participating systems and TCs.
Download
TC
System code
Resultant system
M1
M2
M3
Mn
TC2
TC3
I/O bit field
PB
Address a
Address b
Address n
Memory
System features
Address a
Parameterizations
Address b
System vector
Connectivity vector
I/O ports
Address n
System modes
Host memory
Executable code
TC code

Fig. 122

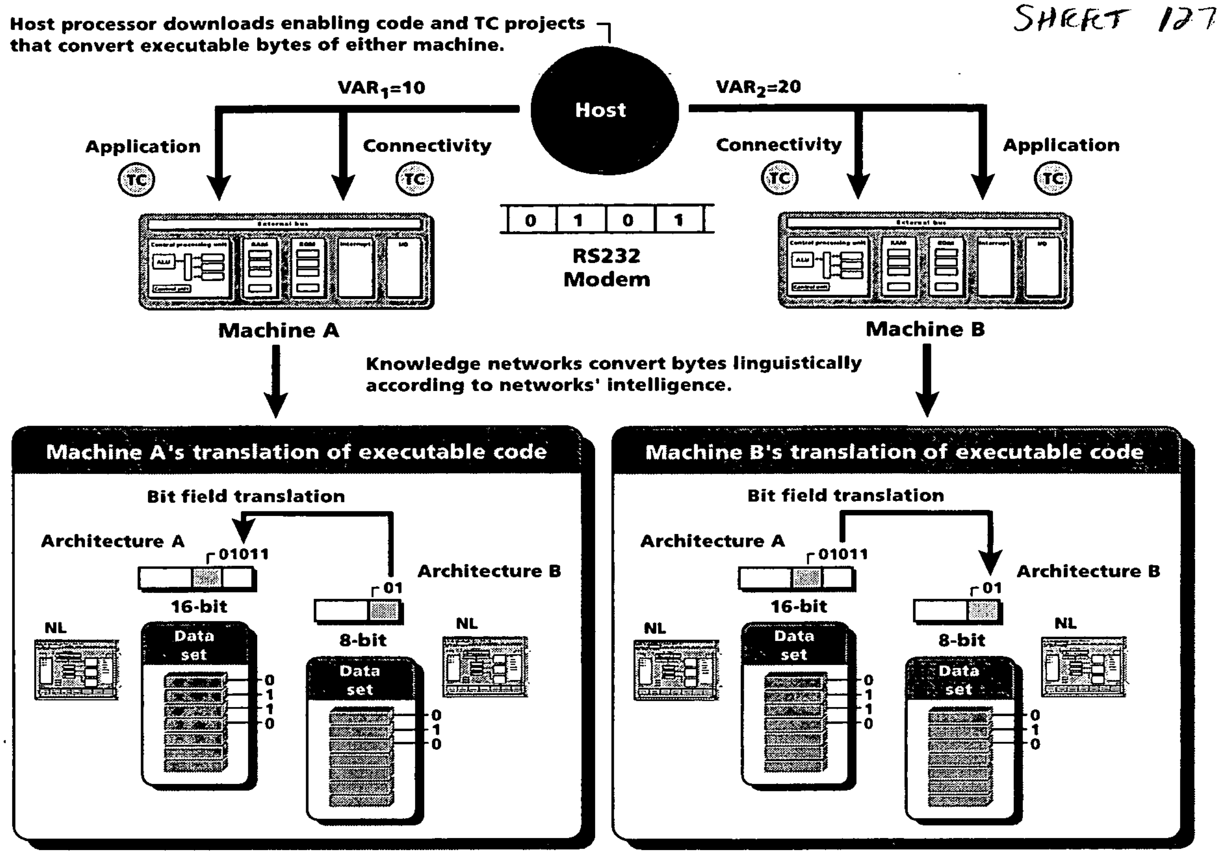

Host processor downloads enabling code and TC projects that convert executable bytes of either machine.
Host
VAR$_1$=10
VAR$_2$=20
Application
Connectivity
TC
Connectivity
Application
TC
External bus
Central processing unit
ALU
Control unit
RAM
ROM
Interrupt
I/O
Machine A
0
1
0
1
RS232 Modem
Machine B
Knowledge networks convert bytes linguistically according to networks' intelligence.
Machine A's translation of executable code
Bit field translation
Architecture A
01011
Architecture B
01
16-bit
8-bit
NL
Data set
0
1
1
0
Data set
0
1
0
NL
Machine B's translation of executable code
Bit field translation
Architecture A
01011
Architecture B
01
16-bit
8-bit
NL
Data set
0
1
1
0
Data set
0
1
0
NL

Fig. 123

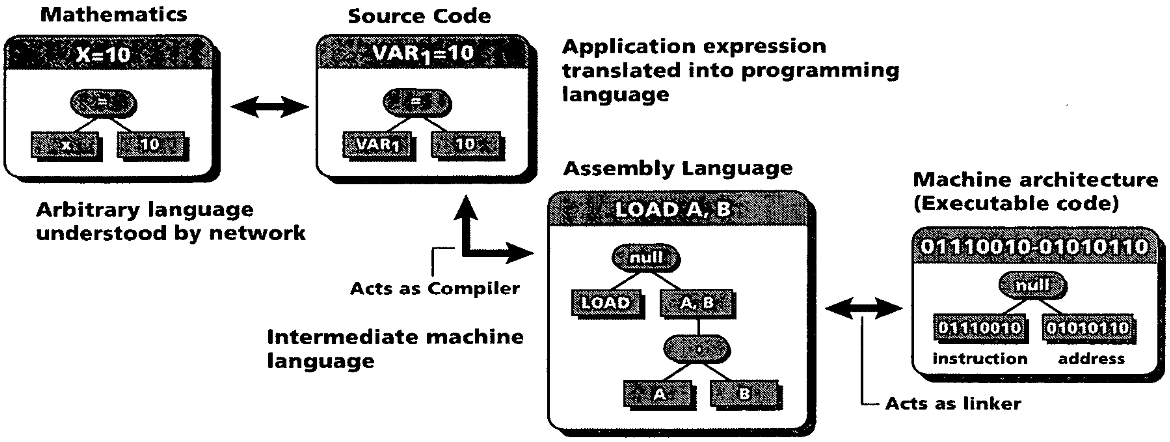

Mathematics
X=10
=
x
10
Arbitrary language understood by network
Source Code
VAR$_1$=10
=
VAR$_1$
10
Application expression translated into programming language
Acts as Compiler
Intermediate machine language
Assembly Language
LOAD A, B
null
LOAD
A, B
,
A
B
Acts as linker
Machine architecture (Executable code)
01110010-01010110
null
01110010
01010110
instruction
address

Fig. 124

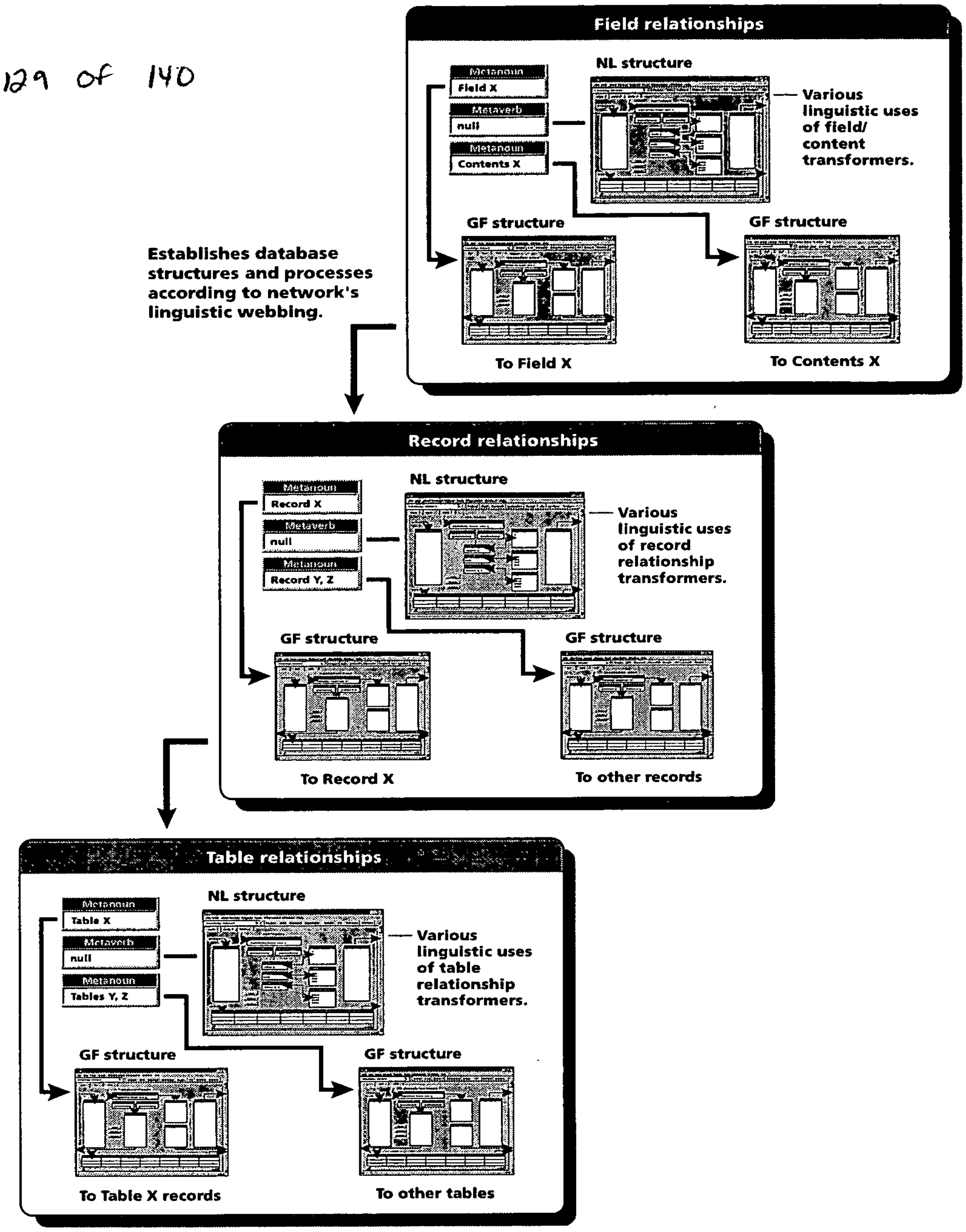
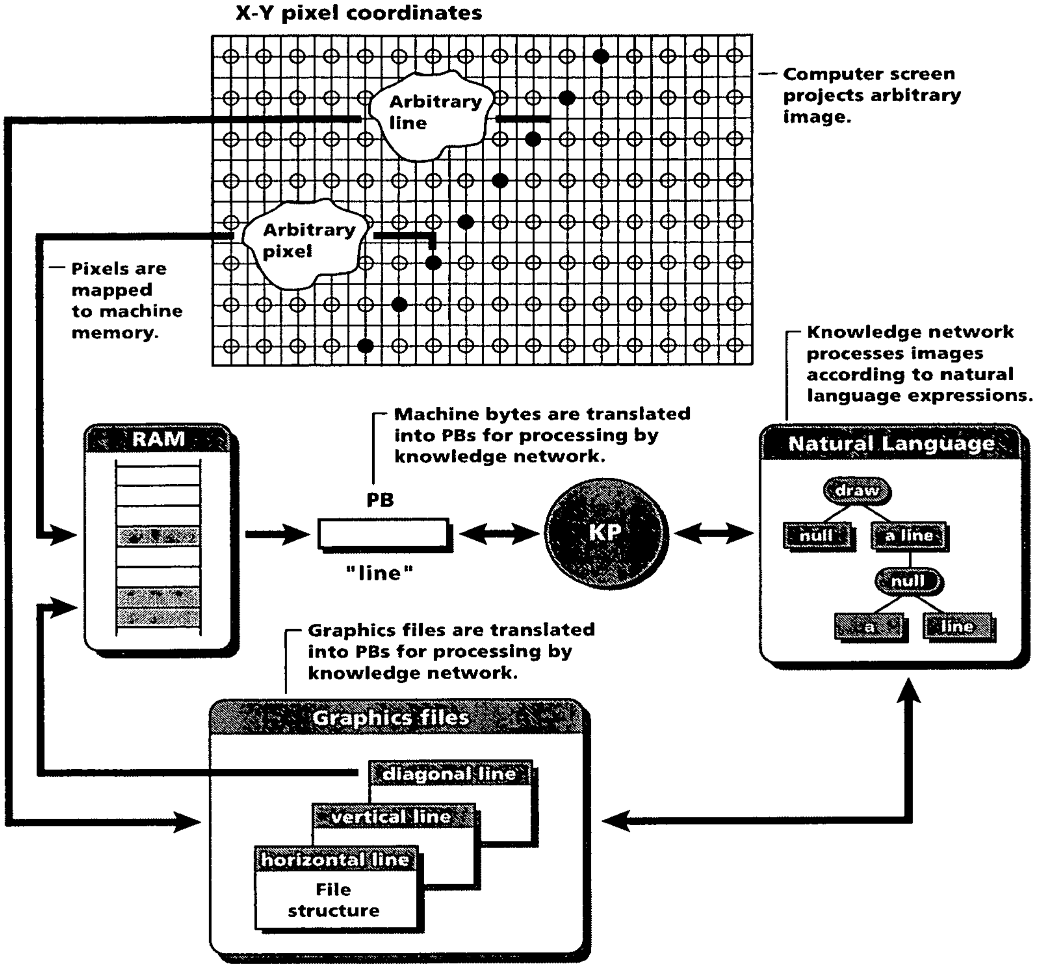

Field relationships
Metanoun
Field X
Metaverb
null
Metanoun
Contents X
NL structure
Various linguistic uses of field/ content transformers.
GF structure
GF structure
To Field X
To Contents X
Establishes database structures and processes according to network's linguistic webbing.
Record relationships
Metanoun
Record X
Metaverb
null
Metanoun
Record Y, Z
NL structure
Various linguistic uses of record relationship transformers.
GF structure
GF structure
To Record X
To other records
Table relationships
Metanoun
Table X
Metaverb
null
Metanoun
Tables Y, Z
NL structure
Various linguistic uses of table relationship transformers.
GF structure
GF structure
To Table X records
To other tables

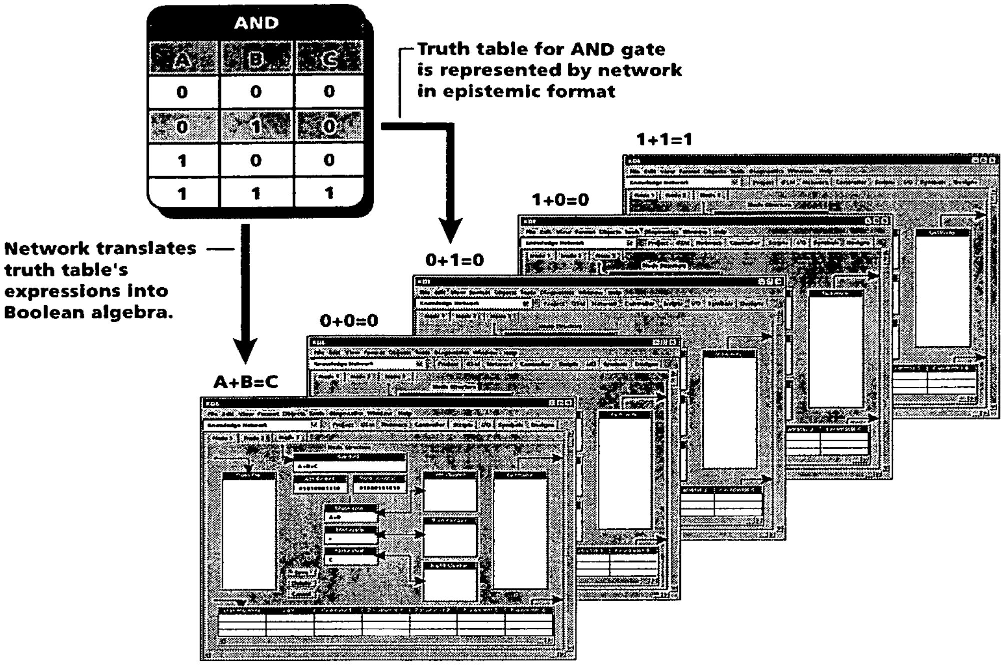
Fig. 125

SHEET 130 OF 140



X-Y pixel coordinates
Computer screen projects arbitrary image.
Arbitrary line
Arbitrary pixel
Pixels are mapped to machine memory.
Knowledge network processes images according to natural language expressions.
Machine bytes are translated into PBs for processing by knowledge network.
RAM
PB
"line"
KP
Natural Language
draw
null
a line
null
a
line
Graphics files are translated into PBs for processing by knowledge network.
Graphics files
diagonal line
vertical line
horizontal line
File structure

Fig. 126



AND		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Truth table for AND gate is represented by network in epistemic format

Network translates truth table's expressions into Boolean algebra.

Fig. 127

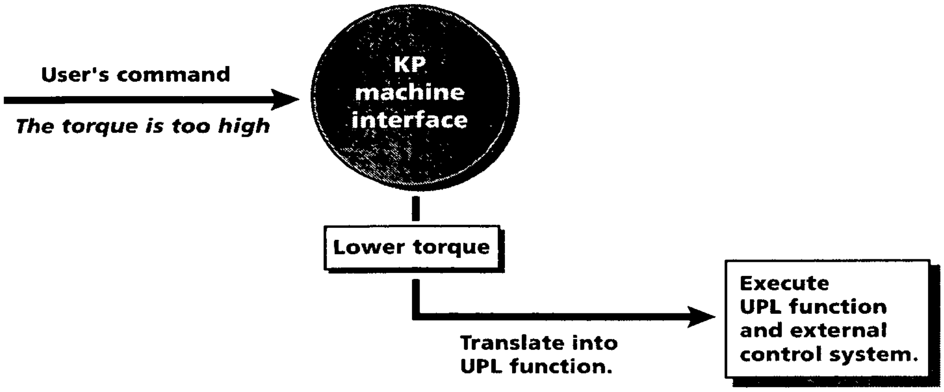

User's command
The torque is too high
KP machine interface
Lower torque
Translate into UPL function.
Execute UPL function and external control system.

Fig. 128

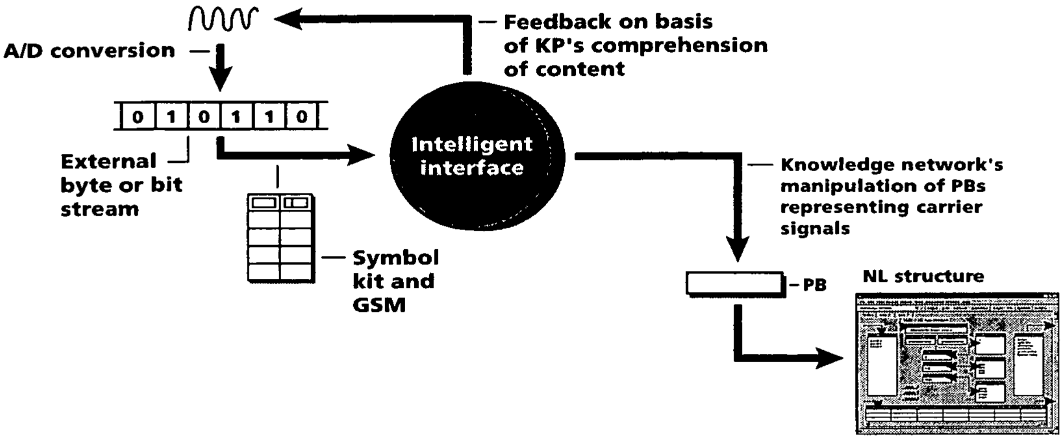

A/D conversion
0 1 0 1 1 0
External byte or bit stream
Symbol kit and GSM
Feedback on basis of KP's comprehension of content
Intelligent interface
Knowledge network's manipulation of PBs representing carrier signals
PB
NL structure

Fig. 129

SHEET 134 OF 140

Simple sentence

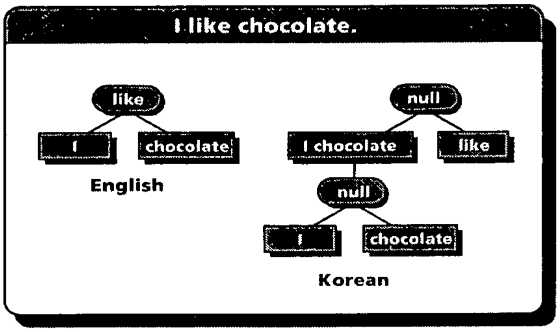


Fig. 130(a)

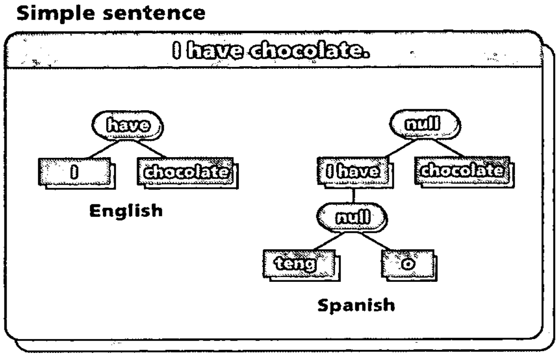

Simple sentence
I have chocolate.
have
I
chocolate
English
null
I have
chocolate
null
teng
o
Spanish

Fig. 130(b)

Apposition

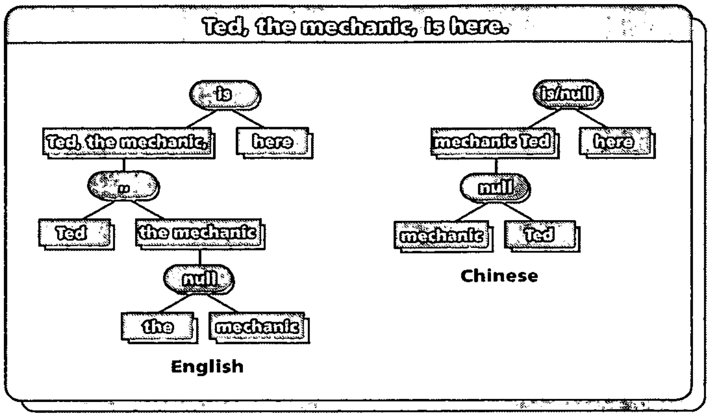
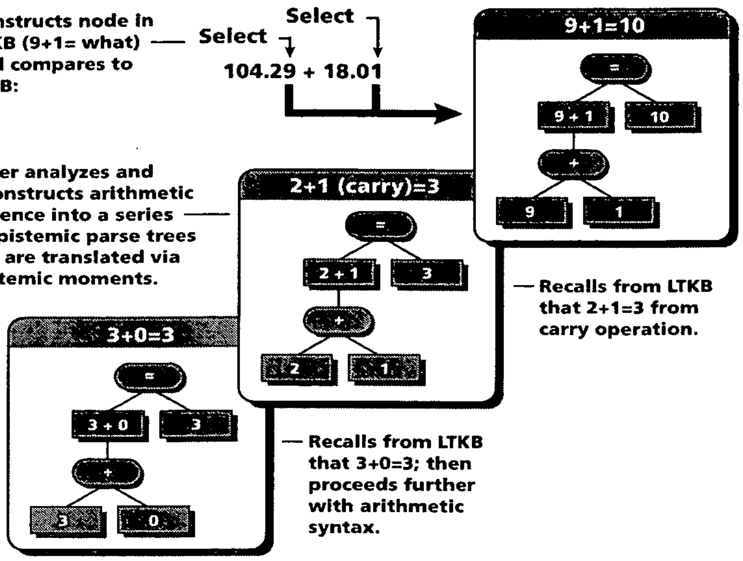


Fig. 130(c)

Constructs node in STKB (9+1= what) and compares to LTKB:



Recalls from LTKB that 9+1=10; then alters ones column.

Parser analyzes and deconstructs arithmetic sentence into a series of epistemic parse trees that are translated via epistemic moments.

Recalls from LTKB that 2+1=3 from carry operation.

Recalls from LTKB that 3+0=3; then proceeds further with arithmetic syntax.

Fig. 131

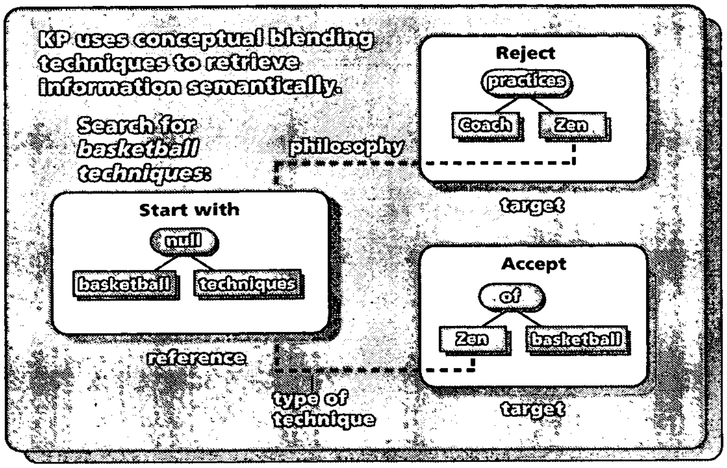
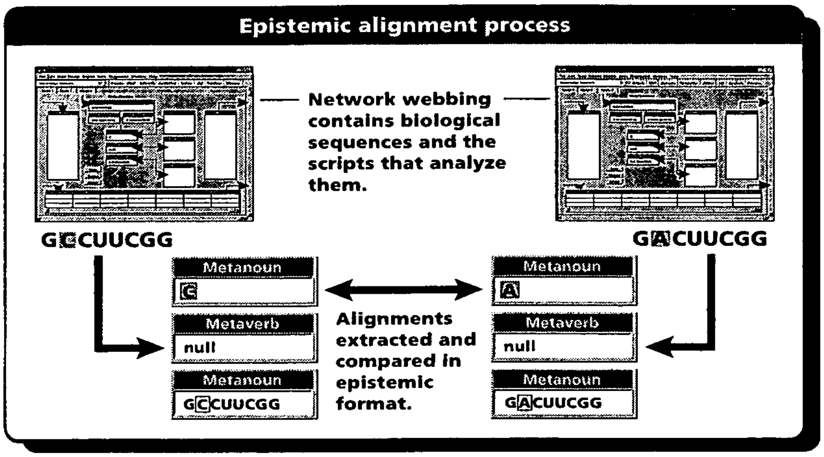

KP uses conceptual blending techniques to retrieve information semantically.
Search for basketball techniques:
Start with
null
basketball
techniques
reference
philosophy
Reject
practices
Coach
Zen
target
type of technique
Accept
of
Zen
basketball
target

Fig. 132



Epistemic alignment process

Network webbing contains biological sequences and the scripts that analyze them.

Alignments extracted and compared in epistemic format.

Fig. 133

SHEET 140 OF 140

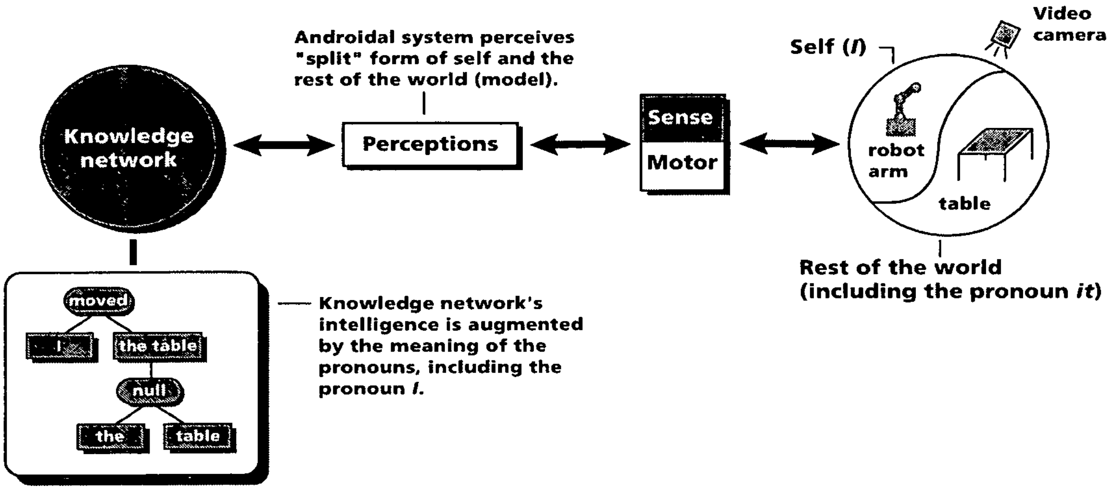

Androidal system perceives "split" form of self and the rest of the world (model).
Knowledge network
Perceptions
Sense
Motor
Self (*I*)
Video camera
robot arm
table
Rest of the world (including the pronoun *it*)
moved
I
the table
null
the
table
Knowledge network's intelligence is augmented by the meaning of the pronouns, including the pronoun *I*.

Fig. 134